In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
import seaborn as sns
```

In [4]:
```
import warnings
warnings.filterwarnings('ignore')
```

1. Data Access

In [5]:
```
# Data Loading
filepath = r"Reporting Analyst Task.csv"

df = pd.read_csv(filepath)
df.head(3)
```

Out[5]:

	id	subject	group_id	assigneeName	status	priority	created_at	updated_at
0	5437	spam	SRC - L1	John brain	open	NaN	Dec 26, 2023, 01:35 AM	Dec 26, 2023, 01:36 AM
1	5436	Infinity - SentinelOne - Black Hash is added/...	SRC - L1	Nir Handa	solved	normal	Dec 26, 2023, 12:57 AM	Dec 26, 2023, 01:00 AM
2	5435	Suspicious email with malicious attachment	SRC - L1	NaN	open	high	Dec 26, 2023, 12:52 AM	Dec 26, 2023, 12:59 AM

In [6]:
```
len(df.columns)
```

Out[6]:
```
15
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5003 entries, 0 to 5002
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              5003 non-null   int64
 1   subject                         5003 non-null   object
 2   group_id                        5003 non-null   object
 3   assigneeName                    4939 non-null   object
 4   status                          5003 non-null   object
 5   priority                        4877 non-null   object
 6   created_at                      5003 non-null   object
 7   updated_at                      5003 non-null   object
 8   channel                         5003 non-null   object
 9   organization_id                 4967 non-null   object
 10  Product                         4419 non-null   object
 11  Alert Severity                  4505 non-null   object
 12  Product Platform                3110 non-null   object
 13  Product Category                4599 non-null   object
 14  Customer First Response (UTC)  502 non-null    object
dtypes: int64(1), object(14)
memory usage: 586.4+ KB
```

2. Data Sanitization

2.1 Updating Column Name & Addessing Missing Values

1. Updates:

- Column Names: Cleaned for consistency, replacing spaces and parentheses with underscores and converting to lowercase.
- Date Columns: Converted created_at, updated_at, and customer_first_response_utc to datetime.

2. Missing Values:

- Some columns have substantial missing values:
- customer_first_response_utc: 90% missing.
- product_platform: 38% missing.
- alert_severity: 10% missing.

In [8]:
```python
# Clean column names for consistency (replace spaces and special characters with
df.columns = df.columns.str.strip().str.replace(" ", "_").str.replace("(", "").s

# Convert date columns to datetime for proper handling
df_date_columns = ["created_at", "updated_at", "customer_first_response_utc"]
for col in df_date_columns:
    df[col] = pd.to_datetime(df[col], errors="coerce")

# Check for missing values in the dataset
missing_summary = df.isnull().sum()

# Display cleaned column names and missing value summary
df.columns
```

Out[8]:
```
Index(['id', 'subject', 'group_id', 'assigneename', 'status', 'priority',
       'created_at', 'updated_at', 'channel', 'organization_id', 'product',
       'alert_severity', 'product_platform', 'product_category',
       'customer_first_response_utc'],
      dtype='object')
```

In [9]:
```
dict(missing_summary)
```

Out[9]:
```
{'id': 0,
 'subject': 0,
 'group_id': 0,
 'assigneename': 64,
 'status': 0,
 'priority': 126,
 'created_at': 0,
 'updated_at': 0,
 'channel': 0,
 'organization_id': 36,
 'product': 584,
 'alert_severity': 498,
 'product_platform': 1893,
 'product_category': 404,
 'customer_first_response_utc': 4506}
```

In [10]:
```
pd.DataFrame(list(missing_summary.items()), columns=['datecolumn', 'Missing Valu
```

Out[10]:

	datecolumn	Missing Values
0	id	0
1	subject	0
2	group_id	0
3	assigneename	64
4	status	0
5	priority	126
6	created_at	0
7	updated_at	0
8	channel	0
9	organization_id	36
10	product	584
11	alert_severity	498
12	product_platform	1893
13	product_category	404
14	customer_first_response_utc	4506

2.2 Handeling Missing Value

- **Filled Missing Values:**

- priority: Replaced missing values with "unknown."
- assigneename and organization_id: Filled with "unassigned" and "unknown," respectively.
- alert_severity, product, product_platform, and product_category: Filled with "not_specified" or "unknown."
- Dropped customer_first_response_utc due to excessive missing data.
- Removed Duplicates: Ensured all records are unique.

In [11]:
```
# Handle missing values based on column significance
# Fill missing `priority` with 'unknown' as it might be categorical
df['priority'].fillna('unknown', inplace=True)

# Fill missing `assigneename` and `organization_id` with 'unassigned' and 'unkno
df['assigneename'].fillna('unassigned', inplace=True)
df['organization_id'].fillna('unknown', inplace=True)

# Drop `customer_first_response_utc` due to excessive missing values (90%)
df.drop(columns=['customer_first_response_utc'], inplace=True)

# Fill missing `alert_severity` and `product` with 'not_specified'
df['alert_severity'].fillna('not_specified', inplace=True)
df['product'].fillna('not_specified', inplace=True)

# Fill `product_platform` and `product_category` with 'unknown' as they may be c
df['product_platform'].fillna('unknown', inplace=True)
df['product_category'].fillna('unknown', inplace=True)

# Remove duplicates if any
df_cleaned = df.drop_duplicates()

# Re-check for missing values and the dataset shape after cleaning
missing_summary_cleaned = df_cleaned.isnull().sum()
df_cleaned.shape, missing_summary_cleaned
```

Out[11]:
```
((5003, 14),
 id                  0
 subject             0
 group_id            0
 assigneename        0
 status              0
 priority            0
 created_at          0
 updated_at          0
 channel             0
 organization_id     0
 product             0
 alert_severity      0
 product_platform    0
 product_category    0
 dtype: int64)
```

In [12]:
```
# Quick summary statistics to identify trends and insights
summary_stats = df_cleaned.describe(include="all").transpose()

# Analyze the distribution of key categorical columns
category_distribution = {
    "status": df_cleaned["status"].value_counts(),
```

```
    "priority": df_cleaned["priority"].value_counts(),
    "alert_severity": df_cleaned["alert_severity"].value_counts(),
    "channel": df_cleaned["channel"].value_counts(),
}

# Display summary stats and distributions
summary_stats
```

Out[12]:

	count	unique	top	freq	mean	
id	5003.0	NaN	NaN	NaN	2712.81731	
subject	5003	2496	Bosch - GuardDuty warning event	122	NaN	
group_id	5003	8	SRC - L1	4712	NaN	
assigneename	5003	29	Gil Kamalakannan	848	NaN	
status	5003	6	closed	4784	NaN	
priority	5003	5	high	2867	NaN	
created_at	5003	NaN	NaN	NaN	2023-06-27 18:22:16.681990656	2 0 04:2
updated_at	5003	NaN	NaN	NaN	2023-07-11 03:27:15.230861568	2 0 00:0
channel	5003	6	web	4933	NaN	
organization_id	5003	40	Bosch	1048	NaN	
product	5003	100	not_specified	584	NaN	
alert_severity	5003	8	medium	2075	NaN	
product_platform	5003	11	unknown	1893	NaN	
product_category	5003	28	cloud_provider_/_service	1402	NaN	

In [13]:

```
category_distribution
```

Out[13]:
```
{'status': status
 closed     4784
 pending     109
 open         60
 solved       26
 new          15
 hold          9
 Name: count, dtype: int64,
 'priority': priority
 high       2867
 normal     1727
 low         222
 unknown     126
 urgent       61
 Name: count, dtype: int64,
 'alert_severity': alert_severity
 medium           2075
 high             1501
 not_specified     498
 critical          423
 low               323
 normal            159
 medium_            17
 urgent              7
 Name: count, dtype: int64,
 'channel': channel
 web                 4933
 api                   48
 email                 13
 side_conversation      7
 voice                  1
 sample_ticket          1
 Name: count, dtype: int64}
```

2.3 Verify Data Types

- Validate numeric columns are not mistakenly stored as strings.

In [14]:
```
df_cleaned.dtypes
```

Out[14]:
```
id                          int64
subject                    object
group_id                   object
assigneename               object
status                     object
priority                   object
created_at          datetime64[ns]
updated_at          datetime64[ns]
channel                    object
organization_id            object
product                    object
alert_severity             object
product_platform           object
product_category           object
dtype: object
```

In [15]:
```
# Convert categorical columns to category data type
categorical_columns = ['status', 'priority', 'alert_severity','channel']
```

```
for col in categorical_columns:
    df_cleaned[col] = df_cleaned[col].astype('category')
```

In [16]:
```
df_cleaned.dtypes
```

Out[16]:
```
id                          int64
subject                    object
group_id                   object
assigneename               object
status                   category
priority                 category
created_at         datetime64[ns]
updated_at         datetime64[ns]
channel                  category
organization_id            object
product                    object
alert_severity           category
product_platform           object
product_category           object
dtype: object
```

2.4 Data Standardization

- Fix inconsistent formatting
- Replacing `medium_` by `medium` value in `alert_severity`
- Replacing "_" with " " from `Channel`
- Capitalised the values of category columns
- Remove unnecessary white spaces or special characters in text columns (`Already Removed`)

In [17]:
```
df_cleaned['alert_severity'].replace('medium_', 'medium', inplace=True)
```

In [18]:
```
df_cleaned['alert_severity'].value_counts()
```

Out[18]:
```
alert_severity
medium           2092
high             1501
not_specified     498
critical          423
low               323
normal            159
urgent              7
Name: count, dtype: int64
```

In [19]:
```
df_cleaned['channel']=df_cleaned['channel'].str.replace('_',' ').str.capitalize(
```

In [20]:
```
df_cleaned['priority']=df_cleaned['priority'].str.replace('_',' ').str.capitaliz
```

In [21]:
```
df_cleaned['status']=df_cleaned['status'].str.replace('_',' ').str.capitalize()
```

In [22]:
```
df_cleaned['product_category']=df_cleaned['product_category'].str.replace('_',' 
```

In [23]:
```
df_cleaned['alert_severity']=df_cleaned['alert_severity'].str.replace('_',' ').s
```

In [24]:
```
df_cleaned['product']=df_cleaned['product'].str.replace('_',' ').str.capitalize(
```

In [25]:
```
df_cleaned.head(5)
```

Out[25]:

	id	subject	group_id	assigneename	status	priority	created_at	updated_
0	5437	spam	SRC - L1	John brain	Open	Unknown	2023-12-26 01:35:00	2023-12-2 01:36:(
1	5436	Infinity - SentinelOne - Black Hash is added/...	SRC - L1	Nir Handa	Solved	Normal	2023-12-26 00:57:00	2023-12-2 01:00:(
2	5435	Suspicious email with malicious attachment	SRC - L1	unassigned	Open	High	2023-12-26 00:52:00	2023-12-2 00:59:(
3	5433	Honda - GCP GKE - User Attempts Multiple Denie...	SRC - L1	Omer Lakhmani	Open	Normal	2023-12-26 00:34:00	2023-12-2 00:35:(
4	5432	Volvo : GuardDuty - Medium Severity Event	SRC - L1	Omer Lakhmani	Pending	Normal	2023-12-25 22:55:00	2023-12-2 22:55:(

2.5 Validate Data Consistency

- Check for valid date ranges (e.g., `updated_at` should not be earlier than `created_at`).
- Verify that ID and other usefull columns (e.g., `organization_id`, `Subject`) have valid entries.

In [26]:
```
df_cleaned['valid_date_range'] = df_cleaned['updated_at'] >= df_cleaned['created

# Step 3: Validate unique and non-null IDs
# Check for duplicates and missing values in `organization_id`
df_cleaned['valid_id'] = df_cleaned['organization_id'].notnull() & ~df_cleaned.d

# Step 4: Filter inconsistent rows (if necessary)
invalid_rows = df_cleaned[~(df_cleaned['valid_date_range'] & df_cleaned['valid_i
```

In [36]:
```
len(invalid_rows)
```

Out[36]:
```
0
```

2.6 Data Exploration for Spam or Test Entries

- Filter records with "spam", "test", and numerical values in `Subject` column

In [28]:
```
# Use regular expressions to identify unwanted rows
pattern = r'\b(spam|test)\b|^\d+'  # Matches "spam", "test", or any numerical va

# Step 2: Remove rows where the Subject column matches the pattern
filtered_df = df_cleaned[~df_cleaned['subject'].str.contains(pattern, case=False
```

In [29]:
```
spam_subject = list(df_cleaned[df_cleaned['subject'].str.contains(r'\b(spam|test
```

In [30]:
```
spam_subject
```

Out[30]:
```
['spam',
 'Test Description to couple of IAM test for GCP CSPM',
 'Re: John - test alert111',
 'NISSAN - CSPM Test - RDS transport encryption enabled',
 'John - test alert111',
 'John - test alert111',
 '360 Learning - Ford - Orca - High Risk Alert',
 'John - test alert111',
 'John - test alert111',
 'John - test alert111',
 'John - test alert111',
 'John - test alert111',
 'John - test alert111',
 'John - test alert111',
 'John - test alert111',
 'John - test alert111',
 'John - test alert111',
 'John - test alert111',
 'John - test alert111',
 'John - test alert111',
 'John - test alert111',
 'test - please dont close!!!',
 'John - test alert111',
 'FOX - test alert4',
 'John - test alert11',
 'John - test alert',
 'John - test alert',
 'John - test alert11',
 'John - test alert11',
 'John - test alert11',
 'John - test alert1',
 'Aeries - CSPM Test',
 'BMW - CSPM Test',
 'Castro - CSPM Test',
 '123123123',
 'KIA - CSPM Test- Critical Checks Failed Report',
 'JupiterMoney - CSPM Test -  Elasticache redis is encrypted at rest',
 'JungleeGames - CSPM Test',
 'Saas - CSPM Test',
 'Angelbroking - CSPM Test',
 'Volvo - CSPM Test - Elasticache cluster is using the latest engine version',
 'Volvo - CSPM Test',
 'Volvo - CSPM Test - KMS key rotation is enabled',
 'Volvo - CSPM Test - RDS is encrypted',
 ' Volvo - CSPM Test - EKS clusters secrets are encrypted',
"Volvo - CSPM Test - S3 Bucket policy doesn't allow global GET action",
 'Honda - CSPM Test - RDS transport encryption enabled',
 'Volvo - CSPM Test - EKS API server is not publicly accessible',
 'Honda - CSPM Test - RDS instance is not publicly accessible',
 'test Jira Integration',
 'Test - Owner - Daniel @ Gmail',
 'test',
 'test',
 'Mazda - Test',
 'Mazda - Test',
 'Mazda - Test',
 'test ',
 'Test Ticket  - Mazda (Hight)',
 'Mazda - Test Alert',
 'Test Jira Integration',
```

```
'Suzuki - CSPM Test - Lambda function is using the latest runtime',
'Suzuki - CSPM Test - Security group restricts inbound Telnet access',
'Suzuki - CSPM Test - EKS API server is not publicly accessible',
'Suzuki - CSPM Test - KMS key rotation is enabled',
'Suzuki - CSPM Test - MFA is enabled for root account',
'Filter Test',
'Suzuki - CSPM Test -  Elasticache redis is encrypted at rest',
'Suzuki - CSPM Test - S3 Bucket is encrypted',
'Suzuki : CSPM Test - EBS volume is attached to an EC2 instance',
'Suzuki - CSPM Test - RDS instance is not publicly accessible',
'Suzuki - CSPM Test - IAM user has MFA enabled for console access',
'test',
'test',
'Mazda : CSPM Test - EBS volume is attached to an EC2 instance',
'Mazda : CSPM Test - is IAM user has MFA enabled for console access',
'Mazda : CSPM Test - Security Group restricts inbound telnet access ',
'test',
'test india',
'test Anna Gilad',
'test ',
'test ticket',
'test proactive ticket1',
'test portal',
"test ticket - don't delete!",
"test public comment - Don't delete!",
'test 123',
'OS - test',
'test']
```

In [31]:
```
filtered_df.shape
```

Out[31]:
```
(4915, 16)
```

In [32]:
```
filtered_df.columns = filtered_df.columns.str.replace('_',' ').str.capitalize()
```

3. Clean Data Extraction

- Prepare visualization charts via Power BI

In [33]:
```
filtered_df.drop(columns=['Valid id','Valid date range'], inplace=True)
```

In [34]:
```
filtered_df.to_csv(r'C:\Users\shailesh.suthar\Downloads\Selu\jupyter\Clean data
```

In [35]:
```
filtered_df.columns
```

Out[35]:
```
Index(['Id', 'Subject', 'Group id', 'Assigneename', 'Status', 'Priority',
       'Created at', 'Updated at', 'Channel', 'Organization id', 'Product',
       'Alert severity', 'Product platform', 'Product category'],
      dtype='object')
```

In [ ]:

In [ ]:

In [ ]: