Cloud Computing Case study

Name	Roll No
Darshit Pithadia	A027
Roshni Panda	A009

What is Cloud Computing?

Cloud computing refers to the delivery of computing services over the internet, allowing users to access software, storage, and processing power without needing to own or maintain physical infrastructure. Instead of running applications or programs from software downloaded on a physical computer or server in their building, users can access cloud-based applications and services from anywhere with an internet connection.

NIST Definition of Cloud Computing

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

Title for Case Study:-

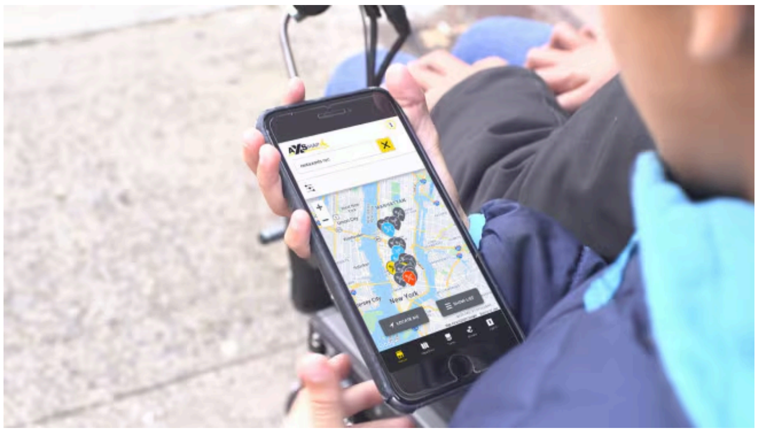
Enhancing AWS Transformation with PwC Solutions

Why is it important in PwC?

Executives are on the hunt for innovative technologies and solutions that can help them future-proof their business. Now, they have their answer: PwC's AWS powered cloud consulting. With AWS, companies can develop large-scale cloud infrastructure quickly - without having to build their own from scratch - while PwC manages the complexities of the implementation. Similarly, AWS can bring other innovative technologies like analytics, blockchain, virtual reality, and Internet of Things while PwC provides the industry know-how to help clients manage their risk, cyber, and data and analytics needs. With PwC's AWS powered cloud services, we can help you prepare your business to adapt and thrive, no matter what the future throws at it.

PwC Small Case Study on Navigating the world with a disability. AXS Map and PwC are changing that.

PwC has worked with companies around the world on their AWS implementations. Here is an example of how they have helped.



Situations/ Background of AXS Map Company
At 25, Jason DaSilva was a rising star in the documentary world. He received numerous awards and even premiered a film at Sundance. Unstoppable, on the rise, and then suddenly slowed by trouble walking. Doctors diagnosed him with an accelerated, and debilitating, form of multiple sclerosis.

Over the next seven years, DaSilva documented his journey in the Emmy Award-winning film When I Walk, and became a disability rights activist. He founded non-profit AXS Lab, Inc., to help people with disabilities through art, media and technology.

AXS Map- Business
Since AXS Map's original 2011 launch, design thinking and human-centered technology advanced, but the web app didn't use current design practices. Nor did it place the end user at the center of strategy and design. They interviewed users with mobility-related disabilities to better understand their frustrations navigating public spaces. There follow-up questions centered around specific pain points in the web app: confusion over the complicated rating system, complex instructions and an unclear purpose.

There’s more. AXS Map could also impact how businesses operate, as many organizations may not know their accessibility shortfalls. As web app traffic expands, businesses will increasingly realize users with mobility related disabilities, their friends and families represent a significant market opportunity. That is, if they pay attention to accessibility. In fact, the US Department of Labor estimates Americans with disabilities spend $175 billion annually on discretionary items and experiences*

How does AXS Map uses AWS

As part of the design process, the AXS Lab and PwC team also reached out to Sasha Blair-Goldensohn, a software engineer on the Google Maps Accessibility team and prominent disability rights advocate for his feedback and experiences. With the new design locked down, PwC’s writers, designers, coders and tech processionals brought the reimagined platform to life.

The team quickly transitioned ratings and location data from the existing AXS Map to the new web app. Then, we shifted our focus to helping reduce costs and better serving unlimited global users by porting AXS Map to Amazon Web Services (AWS). That way, there was no limit to how much the system could scale up or down, depending on the traffic.

It’s easy to forget there are a million little things happening in the background when we use a web app. AXS' platform — hosted on AWS — means a more stable and seamless user experience. AWS is committed to keeping its services up and running approximately 99.9% of the time. That means less crashing, glitching and freezing. As interest in AXS Map grows, the amount of data, resources and information will naturally expand. AWS customers can automatically scale to meet demand within seconds. After the site was completed, the AXS Lab team was able to engage a group of computer science student volunteers to continue development of the site.

Results-A way to more easily navigate daily life

Since AXS Map’s global soft-relaunch in May 2020, the number of searches for accessible places significantly increased. As users add new and updated ratings each day, the number of hassle-free options for individuals with disabilities and their families expands.

Just a few years ago, it was hard for users with mobility-related disabilities to get around without facing multiple unknown obstacles. With advances in technology, design, advocacy and awareness, things continue to change for the better, and AXS Map is a prime example of what can happen when people work together.

AWS Implementation on **Modernising Legacy Applications With Minimal Code(Case Study 1)**

In recent times, companies are focusing more on adopting innovative technologies and are keen to find new, cost-effective solutions. However, old and unsupported technology can act as a barrier in adopting novel solutions due to the risk of breaking the legacy systems which were built with immense time, money and effort
Transitioning from legacy to new applications comes with challenges such as:

- operational downtime
- inadequate security
- slow and inefficient performance
- inability to support growing business needs
- inability to support latest technology integrations
- lack of support for maintenance.

However, the advantages of modernising legacy applications can sometimes outweigh the disadvantages. The key advantages of optimising monolithic architecture of old applications with modern cloud-based solutions are scalability, agility, better performance, security and efficiency.

Databases are one of the most critical and complex components for modernising legacy application's technology stack. PwC India has a mature methodology for designing the legacy database modernisation and in overcoming technical and operational issues associated with modernising legacy databases to be able to leverage cloud-native services. **"PwC is an AWS Premier Tier GSI Partner"** , and the firm also leverages its partnership with AWS to utilise the relevant tools and techniques in this process.

One of the approaches PwC India takes for modernising databases is by adopting fully managed database solutions such as AWS relational database services (RDS) with an open-source engine like PostgreSQL. This helps to reduce the maintenance effort and licensing cost. This approach also increases the reliability and availability of the application. It also enhances the security and reduces the vulnerability as the cloud managed services patches can be applied automatically.

To address these challenges, PwC India identified an approach for database modernisation to Amazon Aurora as a target. This approach combines PwC's database cloud modernisation methodology with a new service from AWS called 'Babelfish for Aurora PostgreSQL'. This service enables Amazon Aurora running on PostgreSQL to understand commands from applications written for Microsoft SQL Server. It also allows the user to run legacy application Microsoft T-SQL code directly on PostgreSQL database engine with minimal modifications.

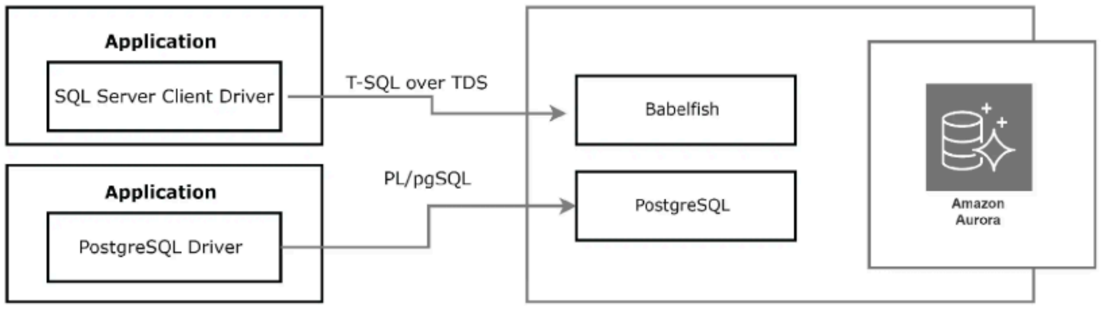
Introduction to Amazon Aurora

Amazon Aurora (Aurora) is a fully managed relational database engine (RDBMS) that's fully compatible with MySQL and PostgreSQL. Aurora gives you the performance and availability of commercial-grade databases at one-tenth the cost. This cloud-native database platform includes high-performance features including serverless options, auto-scaling storage, and the ability to have up to 15 low-latency read replicas and to run globally distributed applications.

Now Let's **Introduce BabelFish**

Babelfish for Aurora PostgreSQL is a new capability for Amazon Aurora PostgreSQL – compatible edition that enables Aurora to understand commands from applications written for a Microsoft SQL server.

Babelfish is designed to provide an additional endpoint for a PostgreSQL database cluster to understand the SQL server wire-level protocol and commonly used SQL server statements (T-SQL). This allows client applications that use the tabular data stream (TDS) wire protocol to connect natively to the TDS listener port on PostgreSQL. It can accept incoming connections from one application using Babelfish TDS and another application's native PostgreSQL connection at the same time.1



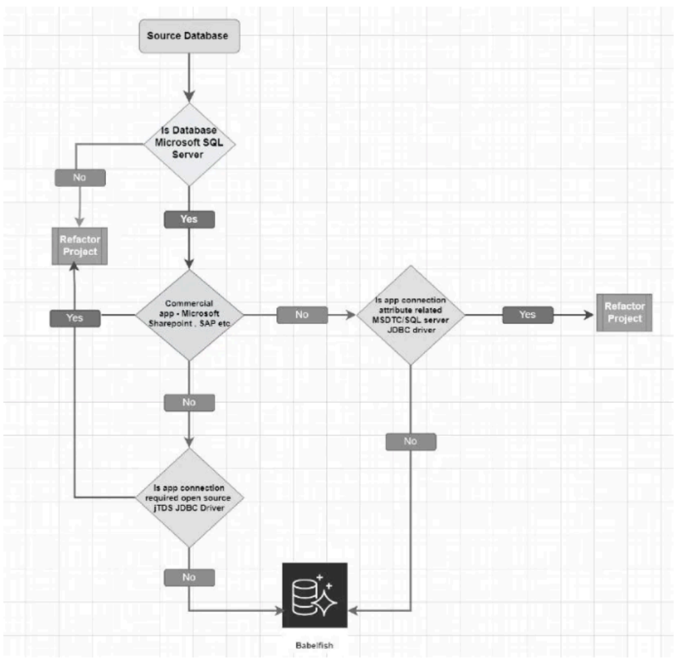
PwC’s approach for MSSQL applications modernisation to AWS cloud with Babelfish

Babelfish allows the application to continue using T-SQL (Microsoft SQL Server’s proprietary SQL dialect). It also supports native PostgreSQL connections so that the application can be modernised at the same time as the database if the customer wishes to do so.

In this way, Babelfish for Aurora PostgreSQL reduces the technical issues of moving to a different database engine. Customers gain better scalability and save the cost by migrating to Amazon Aurora, without all the refactoring work that traditionally is needed.

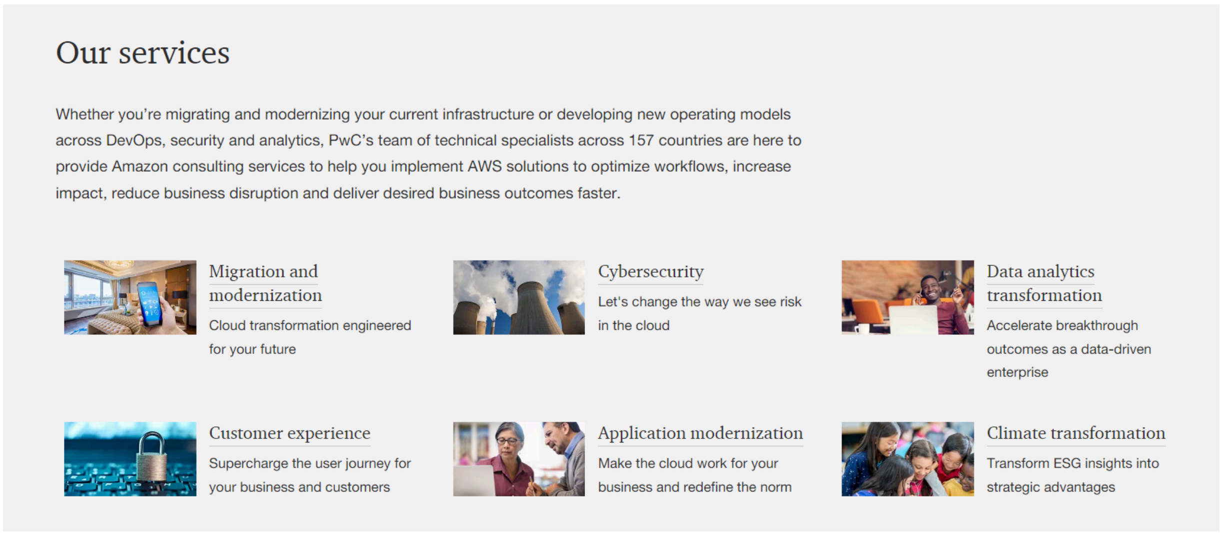
Assessment | Proof of concept (PoC) | Migrate and modernise

PwC conducts the assessment using the decision tree method shown below to analyse the dependency of database engine and take the decision whether it can include the application for modernisation or not.


Source Database
Is Database Microsoft SQL Server
No
Refactor Project
Yes
Commercial app - Microsoft Sharepoint , SAP etc
Yes
No
Is app connection attribute related MSDTC/SQL server JDBC driver
Yes
Refactor Project
No
No
Is app connection required open source jTDS JDBC Driver
No
Babelfish

This is not a Case Study but how AWS can be implemented by Modernising Legacy Applications by writing Minimal Lines of Code.

Services Offered by PWC



Our services

Whether you're migrating and modernizing your current infrastructure or developing new operating models across DevOps, security and analytics, PwC's team of technical specialists across 157 countries are here to provide Amazon consulting services to help you implement AWS solutions to optimize workflows, increase impact, reduce business disruption and deliver desired business outcomes faster.

Migration and modernization
Cloud transformation engineered for your future

Cybersecurity
Let's change the way we see risk in the cloud

Data analytics transformation
Accelerate breakthrough outcomes as a data-driven enterprise

Customer experience
Supercharge the user journey for your business and customers

Application modernization
Make the cloud work for your business and redefine the norm

Climate transformation
Transform ESG insights into strategic advantages

Cloud Services- Data and Analytics

Now, more than ever, is the time to consider how to look at the data your company is gathering. Are you optimising your assets and making better, faster decisions? Are you able to work more efficiently and save money? Are you identifying new sources of revenue to tap into potential markets? It's time to let PwC and AWS help propel your data strategy to new heights.

PwC solutions for data and analytics in AWS

- Enhance your **employees' productivity** through **automation** — while **maintaining quality** and compliance — and helping to **reduce costs**.
- Create impactful **customer experiences** with **AI tools** that can proactively anticipate their needs based on their preferences.
- Elevate the engineering prowess of your organisation to help get the most out of your **investment in cloud**, **increase performance** of your cloud tools and innovate **new products and services**.

Problem Statement(Case Study 2):-

How Wyndham Hotels & Resorts improved their data capabilities to get to know their guests better

Introduction

Running one hotel isn't easy - running 9,000 hotels across 80 countries is a challenge on a different scale

In 2020, Wyndham Hotels and Resorts, a franchisor which operates brands ranging from Super 8 to Wyndham Grand, was four years into a "cloud first" strategy and migration that had seen some early successes. Still, opportunities remained. Like many hospitality enterprises, Wyndham has grown in large part by acquisition to become a major international hotelier spanning 20 separate brands processing more than a million reservations a day.

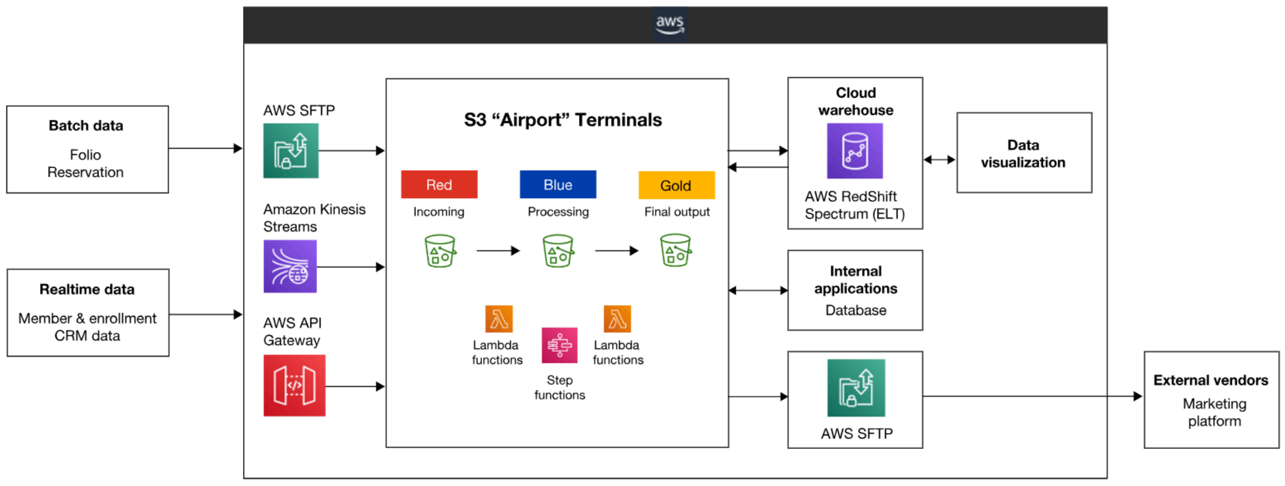
Wyndham knew that data could provide a better understanding of guests, their stays and their preferences at Wyndham properties—and thus help positively impact the overall guest experience and loyalty. In 2016, the company undertook a large-scale data architecture overhaul as a first step. However, despite the years of investment, Wyndham still had opportunities for maximising the potential of its data assets and to derive further benefits from investments already made.

Goal: Manage the Wyndham business more effectively by redesigning the data landscape to glean further insights and connect with guests in a personalised manner.

Wyndham had already implemented Amazon Web Services (AWS) as the basis for a new, cloud-based back end for data processing. However it was still relying on legacy platforms to ingest, move and process data. The first step was to help Wyndham more fully utilise AWS tools and services, while standardising the methods by which data was transferred between its core systems. Specifically, Wyndham's move to a more robust, scalable cloud environment provided improved quality checks, better error controls and better overall accuracy of its guest data.

One significant challenge was revamping the way that Wyndham handled data, including reviews for data integrity requiring manual review of each error—a painstakingly slow and costly process. In the **new data architecture**, numerous AWS Platform as a Service (PaaS) tools were used, such as **Amazon Kinesis**, **SFTP**, **Amazon API Gateway**, to make sure that Wyndham's data was streamlined, automated and moved to the new AWS environment securely. From an operational standpoint, tools such as Amazon CloudWatch, Amazon Elasticsearch and Amazon QuickSight enabled Wyndham to quickly visualise guest and property metrics through interactive dashboards. In aggregate, the newly established architecture – packaged using AWS Serverless Application Model (SAM) - allowed Wyndham to access guest insights at a depth and speed which they had never experienced before.

A look of Wyndham's new data architecture on AWS



The term Airport is used as a metaphor where data is centralized and stored in one place

Results on Case Study 1

Before PwC entered the engagement, a single piece of data would travel an average of nine system steps to get from its point of creation to Wyndham's central database, an error-fraught trip which would take up to two-and-a-half days to complete. Today, that number of steps has been reduced to four, and the journey now takes as little as five minutes. The quality of data has also been improved because it has been so effectively centralised, reducing the likelihood of inconsistencies and conflicts.
Wyndham has also seen an estimated 40 percent decrease in the time spent on managing its computing environment, thanks to being able to fully leverage the serverless architecture enabled through AWS. This in turn has helped reduce cost for

the company, while now allowing Wyndham’s properties and corporate team to develop a fuller, more accurate understanding of its guests and their needs.
While the COVID-19 pandemic has slowed the travel industry in general, Wyndham is well-positioned for the impending recovery, thanks to a more efficient and accurate data architecture, rewarding the hotelier to help its owners spend more time focusing on their guests and less time investigating root causes behind data anomalies or inconsistencies.

Problem Statement Case Study 3:-
How PwC reimagines fusion centre powered by Amazon Security Lake

Introduction to PwC’s

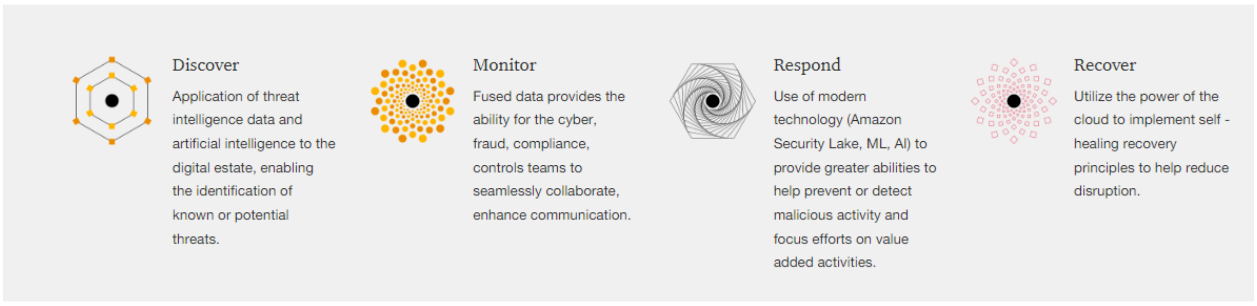
PwC’s Cyber, Risk, and Regulatory Practice brings knowledge and expertise to aid clients in implementing the fusion centre to meet their individual needs. Built on AWS, the fusion centre provides the ability to combine data from a variety of sources to create a centralised, near real-time view. The dashboard features visualisations, drill-down capabilities, and automated workflows, enabling teams to quickly identify, investigate, and respond to potential risks posed by the ever evolving threat actors.

What is Fusion Centre??

Fusion centre is a cutting edge customizable and scalable architecture composed of multiple layers that work together to connect, secure, and safeguard an organisation's environment. This architecture can provide an organisation the ability to leverage the capabilities of AWS security lake to monitor, respond, and recover from cyber, compliance, and fraud events in a more effective and efficient manner harnessing the power of AI/ML.

What are the benefits of the fusion centre?

1. Discover - Application of threat Intelligence data an Artificial Intelligence to the digital estate, enabling identification
2. Monitor- Continuously Monitor Fusion data providers for Fraud and other conscious activities to be taken care of
3. Respond- The Amazon Security Lake provides greater abilities for ML and AI



Discover

Application of threat intelligence data and artificial intelligence to the digital estate, enabling the identification of known or potential threats.

Monitor

Fused data provides the ability for the cyber, fraud, compliance, controls teams to seamlessly collaborate, enhance communication.

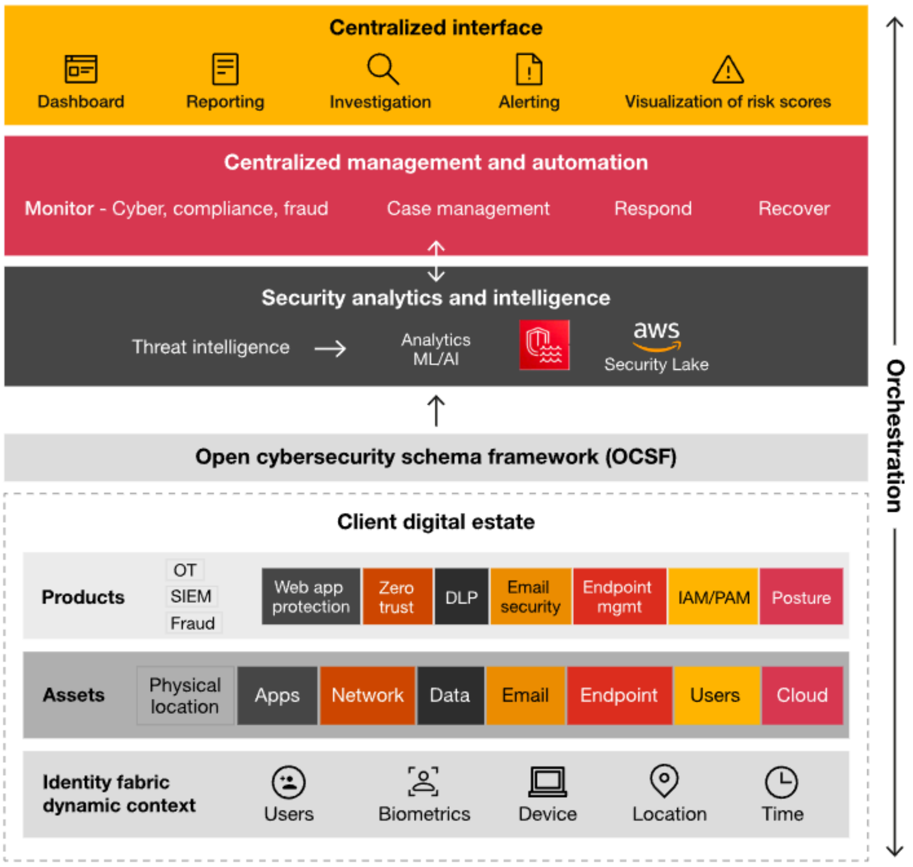
Respond

Use of modern technology (Amazon Security Lake, ML, AI) to provide greater abilities to help prevent or detect malicious activity and focus efforts on value added activities.

Recover

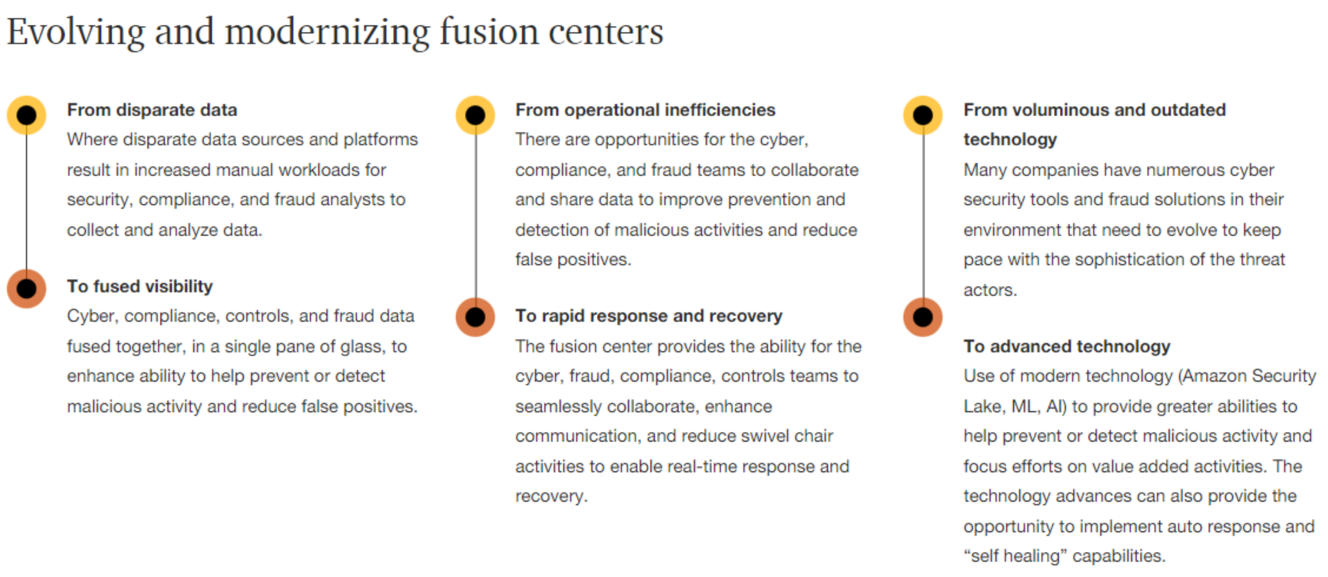
Utilize the power of the cloud to implement self - healing recovery principles to help reduce disruption.

Example architecture of the fusion centre built on AWS Security Lake



Reimagining fusion centres

PwC's Cyber, Risk, and Regulatory practice brings knowledge and expertise to aid clients in implementing the fusion centre to meet their individual needs. Built on AWS, the fusion centre provides the ability to combine data from a variety of sources to create a centralised, near real-time view. The dashboard features visualisations, drill-down capabilities, and automated workflows, enabling teams to quickly identify, investigate, and respond to potential risks posed by the ever evolving threat actors.



Evolving and modernizing fusion centers

From disparate data
Where disparate data sources and platforms result in increased manual workloads for security, compliance, and fraud analysts to collect and analyze data.

To fused visibility
Cyber, compliance, controls, and fraud data fused together, in a single pane of glass, to enhance ability to help prevent or detect malicious activity and reduce false positives.

From operational inefficiencies
There are opportunities for the cyber, compliance, and fraud teams to collaborate and share data to improve prevention and detection of malicious activities and reduce false positives.

To rapid response and recovery
The fusion center provides the ability for the cyber, fraud, compliance, controls teams to seamlessly collaborate, enhance communication, and reduce swivel chair activities to enable real-time response and recovery.

From voluminous and outdated technology
Many companies have numerous cyber security tools and fraud solutions in their environment that need to evolve to keep pace with the sophistication of the threat actors.

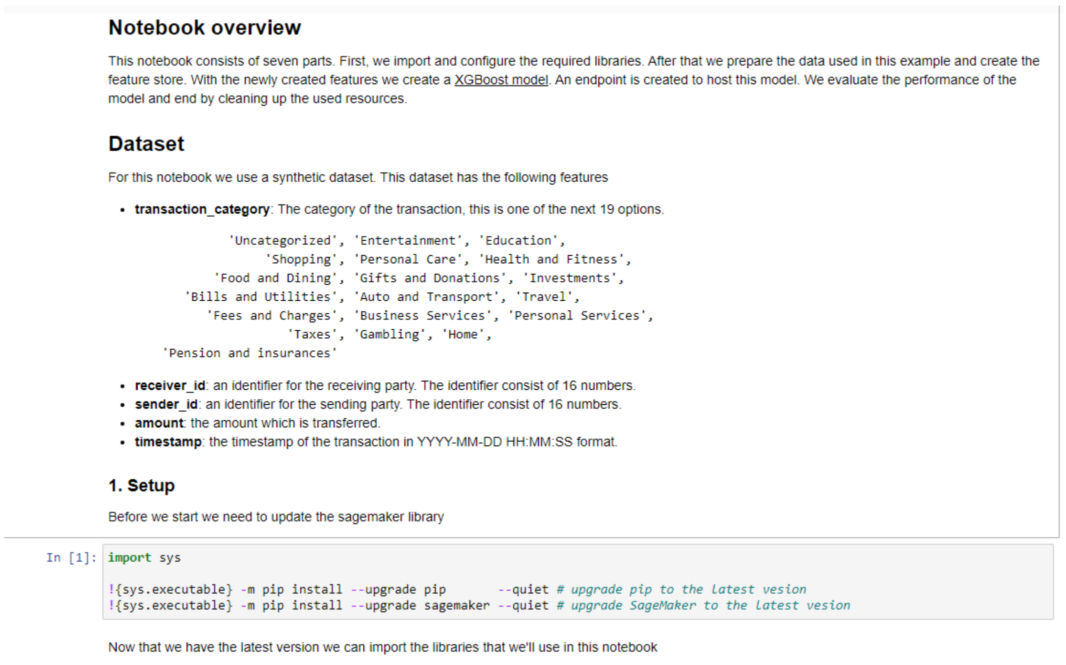
To advanced technology
Use of modern technology (Amazon Security Lake, ML, AI) to provide greater abilities to help prevent or detect malicious activity and focus efforts on value added activities. The technology advances can also provide the opportunity to implement auto response and "self healing" capabilities.

Conclusion

In conclusion, cloud computing revolutionises how computing services are delivered, providing users with convenient access to resources over the internet. It eliminates the need for owning physical infrastructure and enables rapid provisioning and release of resources with minimal management effort. PwC's adoption of AWS exemplifies the strategic utilisation of cloud technology to enhance business operations, drive innovation, and deliver value-added solutions to clients. By leveraging AWS, PwC can efficiently scale services, ensure security and compliance, and facilitate global collaboration, ultimately empowering businesses to adapt and thrive in a rapidly evolving digital landscape.

Deploying Machine Learning Algorithm on Amazon SageMaker

In this we will use XGBoost Model to Classify the Payment Transactions.
In this the Notebook it consists of the Complete End-to-End Payment Classification Technique which is done using XGBoost Technique. However we tried to deploy the model but the data was not able to get uploaded on the S3 Buckets.



Notebook overview

This notebook consists of seven parts. First, we import and configure the required libraries. After that we prepare the data used in this example and create the feature store. With the newly created features we create a XGBoost model. An endpoint is created to host this model. We evaluate the performance of the model and end by cleaning up the used resources.

Dataset

For this notebook we use a synthetic dataset. This dataset has the following features

- **transaction_category**: The category of the transaction, this is one of the next 19 options.

```
                'Uncategorized', 'Entertainment', 'Education',
                     'Shopping', 'Personal Care', 'Health and Fitness',
              'Food and Dining', 'Gifts and Donations', 'Investments',
          'Bills and Utilities', 'Auto and Transport', 'Travel',
             'Fees and Charges', 'Business Services', 'Personal Services',
                        'Taxes', 'Gambling', 'Home',
       'Pension and insurances'
```

- **receiver_id**: an identifier for the receiving party. The identifier consist of 16 numbers.
- **sender_id**: an identifier for the sending party. The identifier consist of 16 numbers.
- **amount**: the amount which is transferred.
- **timestamp**: the timestamp of the transaction in YYYY-MM-DD HH:MM:SS format.

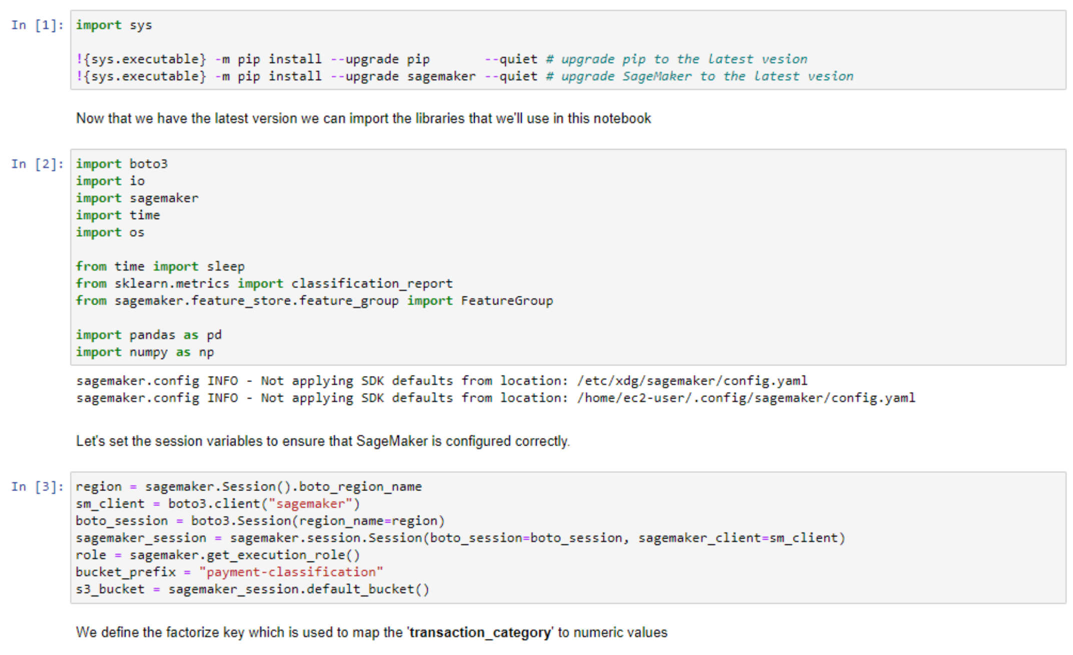
1. Setup

Before we start we need to update the sagemaker library

```
In [1]: import sys

        !{sys.executable} -m pip install --upgrade pip       --quiet # upgrade pip to the latest vesion
        !{sys.executable} -m pip install --upgrade sagemaker --quiet # upgrade SageMaker to the latest vesion
```

Now that we have the latest version we can import the libraries that we'll use in this notebook

Here importing multiple libraries, which consist of Sagemaker Library through which a Session is created and it creates Feature Group which allows for grouping the data and passing on to S3 Buckets



```
In [1]: import sys

        !{sys.executable} -m pip install --upgrade pip       --quiet # upgrade pip to the latest vesion
        !{sys.executable} -m pip install --upgrade sagemaker --quiet # upgrade SageMaker to the latest vesion
```

Now that we have the latest version we can import the libraries that we'll use in this notebook

```
In [2]: import boto3
        import io
        import sagemaker
        import time
        import os

        from time import sleep
        from sklearn.metrics import classification_report
        from sagemaker.feature_store.feature_group import FeatureGroup

        import pandas as pd
        import numpy as np
```

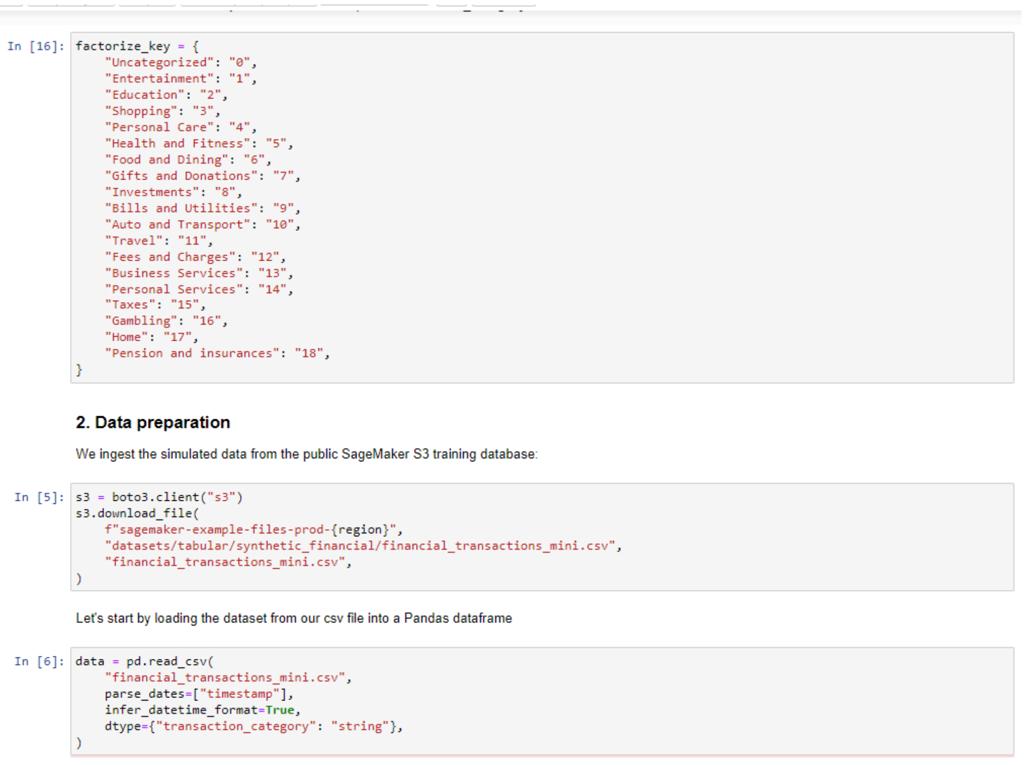
```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-user/.config/sagemaker/config.yaml
```

Let's set the session variables to ensure that SageMaker is configured correctly.

```
In [3]: region = sagemaker.Session().boto_region_name
        sm_client = boto3.client("sagemaker")
        boto_session = boto3.Session(region_name=region)
        sagemaker_session = sagemaker.session.Session(boto_session=boto_session, sagemaker_client=sm_client)
        role = sagemaker.get_execution_role()
        bucket_prefix = "payment-classification"
        s3_bucket = sagemaker_session.default_bucket()
```

We define the factorize key which is used to map the '**transaction_category**' to numeric values

Now that Multiple libraries are imported our data needs to be prepared which includes Data Manipulation and other techniques which helps transforming the Data and can apply various Models for Deployment.



```
In [16]: factorize_key = {
             "Uncategorized": "0",
             "Entertainment": "1",
             "Education": "2",
             "Shopping": "3",
             "Personal Care": "4",
             "Health and Fitness": "5",
             "Food and Dining": "6",
             "Gifts and Donations": "7",
             "Investments": "8",
             "Bills and Utilities": "9",
             "Auto and Transport": "10",
             "Travel": "11",
             "Fees and Charges": "12",
             "Business Services": "13",
             "Personal Services": "14",
             "Taxes": "15",
             "Gambling": "16",
             "Home": "17",
             "Pension and insurances": "18",
         }
```

2. Data preparation

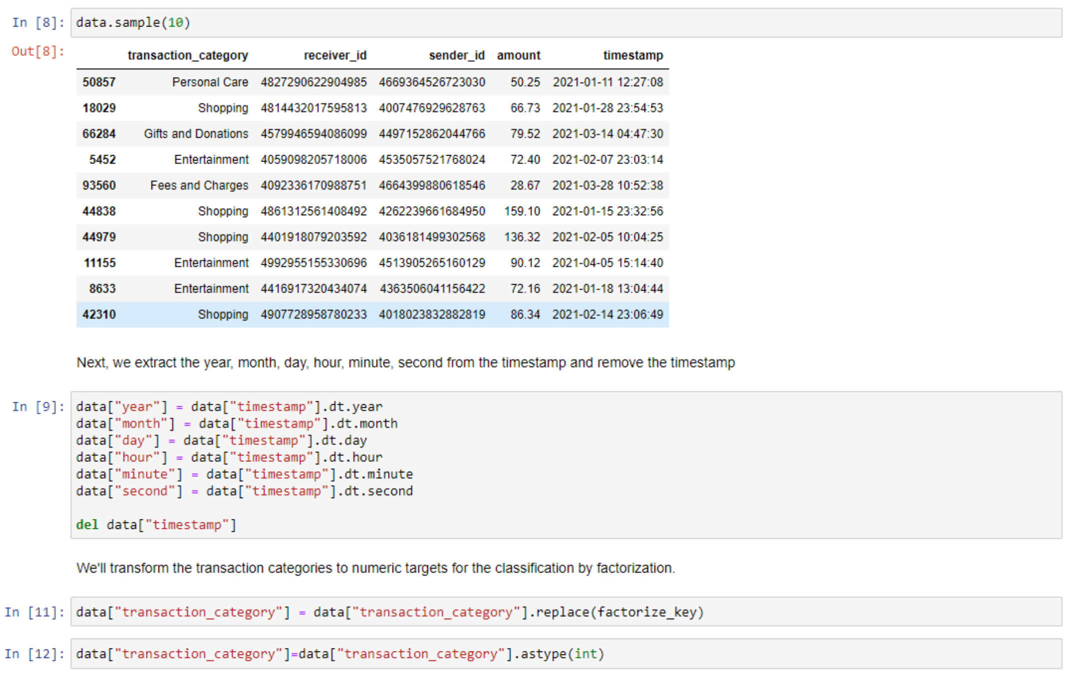
We ingest the simulated data from the public SageMaker S3 training database:

```
In [5]: s3 = boto3.client("s3")
        s3.download_file(
            f"sagemaker-example-files-prod-{region}",
            "datasets/tabular/synthetic_financial/financial_transactions_mini.csv",
            "financial_transactions_mini.csv",
        )
```

Let's start by loading the dataset from our csv file into a Pandas dataframe

```
In [6]: data = pd.read_csv(
            "financial_transactions_mini.csv",
            parse_dates=["timestamp"],
            infer_datetime_format=True,
            dtype={"transaction_category": "string"},
        )
```

Checking that the data is loaded with Sample and then convert Into Necessary Datatypes which required for Models to be passed on as input. Also applying Encoding Techniques for Model understanding the Model understands Quantitative Data rather than Qualitative Data.


In [8]:
```
data.sample(10)
```
Out[8]:

	transaction_category	receiver_id	sender_id	amount	timestamp
50857	Personal Care	4827290622904985	4669364526723030	50.25	2021-01-11 12:27:08
18029	Shopping	4814432017595813	4007476929628763	66.73	2021-01-28 23:54:53
66284	Gifts and Donations	4579946594086099	4497152862044766	79.52	2021-03-14 04:47:30
5452	Entertainment	4059098205718006	4535057521768024	72.40	2021-02-07 23:03:14
93560	Fees and Charges	4092336170988751	4664399880618546	28.67	2021-03-28 10:52:38
44838	Shopping	4861312561408492	4262239661684950	159.10	2021-01-15 23:32:56
44979	Shopping	4401918079203592	4036181499302568	136.32	2021-02-05 10:04:25
11155	Entertainment	4992955155330696	4513905265160129	90.12	2021-04-05 15:14:40
8633	Entertainment	4416917320434074	4363506041156422	72.16	2021-01-18 13:04:44
42310	Shopping	4907728958780233	4018023832882819	86.34	2021-02-14 23:06:49

Next, we extract the year, month, day, hour, minute, second from the timestamp and remove the timestamp

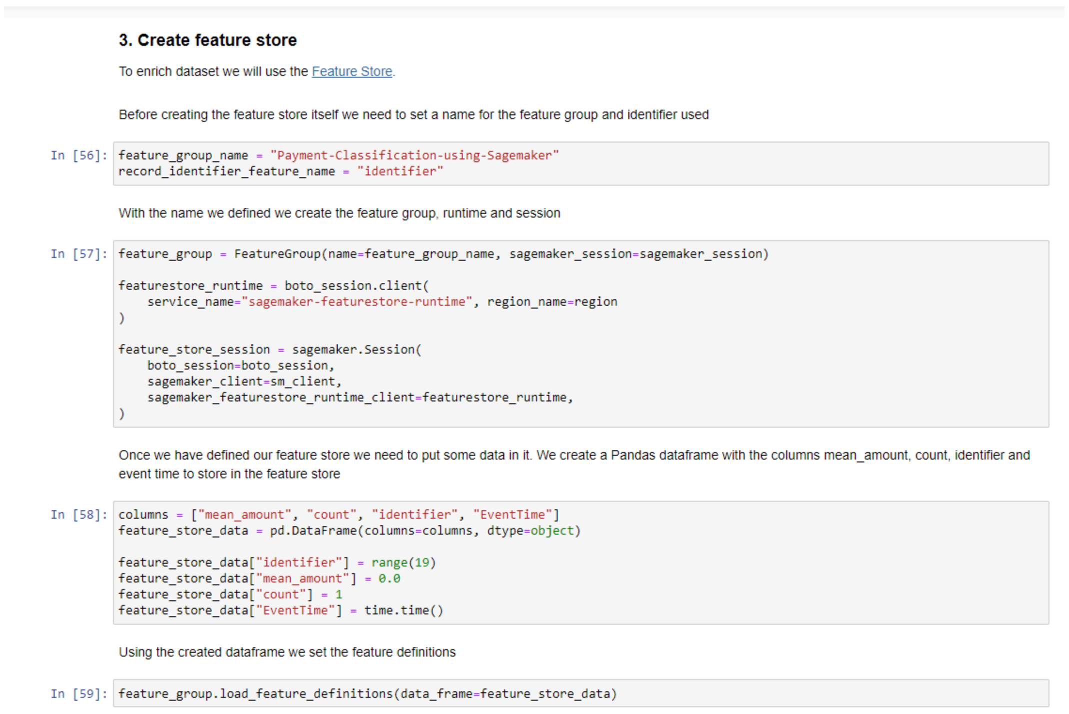
In [9]:
```
data["year"] = data["timestamp"].dt.year
data["month"] = data["timestamp"].dt.month
data["day"] = data["timestamp"].dt.day
data["hour"] = data["timestamp"].dt.hour
data["minute"] = data["timestamp"].dt.minute
data["second"] = data["timestamp"].dt.second

del data["timestamp"]
```

We'll transform the transaction categories to numeric targets for the classification by factorization.

In [11]:
```
data["transaction_category"] = data["transaction_category"].replace(factorize_key)
```

In [12]:
```
data["transaction_category"]=data["transaction_category"].astype(int)
```

Now in this below Flow it has been observed that a Feature Store is created from the Dataset links provided and an Sagemaker Session is created using the region and the unique name is assigned to call the SageMaker Services.


3. Create feature store

To enrich dataset we will use the Feature Store.

Before creating the feature store itself we need to set a name for the feature group and identifier used

In [56]:
```
feature_group_name = "Payment-Classification-using-Sagemaker"
record_identifier_feature_name = "identifier"
```

With the name we defined we create the feature group, runtime and session

In [57]:
```
feature_group = FeatureGroup(name=feature_group_name, sagemaker_session=sagemaker_session)

featurestore_runtime = boto_session.client(
    service_name="sagemaker-featurestore-runtime", region_name=region
)

feature_store_session = sagemaker.Session(
    boto_session=boto_session,
    sagemaker_client=sm_client,
    sagemaker_featurestore_runtime_client=featurestore_runtime,
)
```

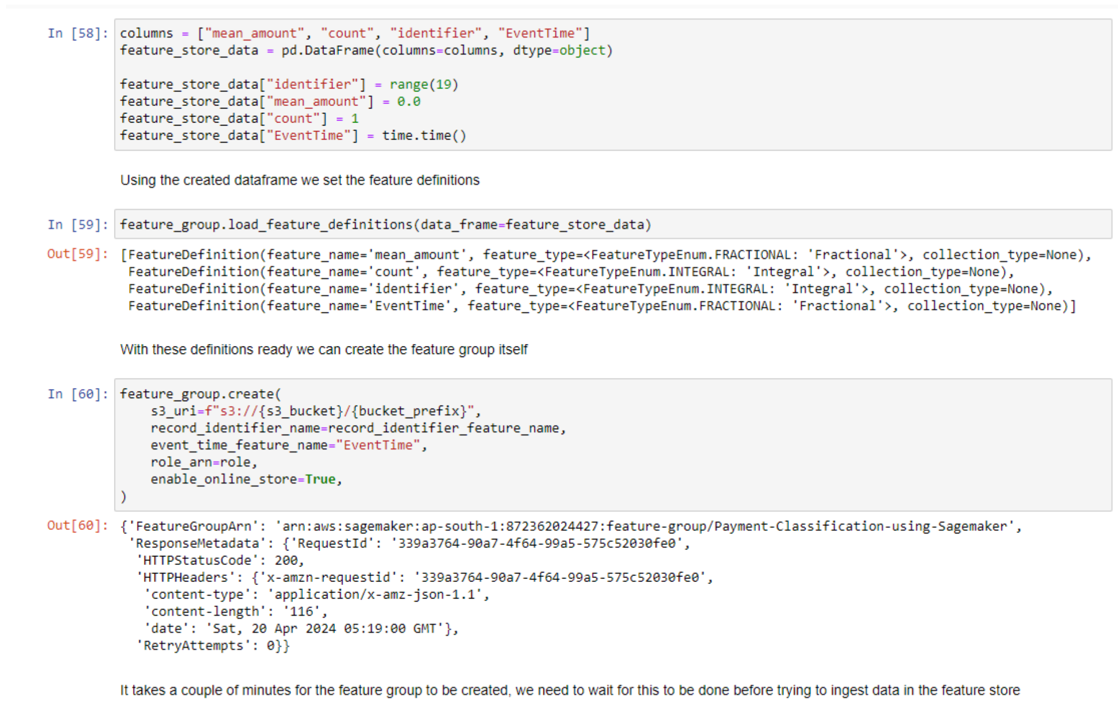
Once we have defined our feature store we need to put some data in it. We create a Pandas dataframe with the columns mean_amount, count, identifier and event time to store in the feature store

In [58]:
```
columns = ["mean_amount", "count", "identifier", "EventTime"]
feature_store_data = pd.DataFrame(columns=columns, dtype=object)

feature_store_data["identifier"] = range(19)
feature_store_data["mean_amount"] = 0.0
feature_store_data["count"] = 1
feature_store_data["EventTime"] = time.time()
```

Using the created dataframe we set the feature definitions

In [59]:
```
feature_group.load_feature_definitions(data_frame=feature_store_data)
```

As the various Features are extracted from the above code and feature group name is used to collect those Feature into group called as Feature Data Store and further Feature Definitions are used where the store data . With these Feature we can create the Definitions itself and now this Feature is loaded into an S3 Bucket by mentioning the unique identifier Name and enabling various Roles and Services assigned to it by the IAM user Services and Policies.



```
In [58]: columns = ["mean_amount", "count", "identifier", "EventTime"]
         feature_store_data = pd.DataFrame(columns=columns, dtype=object)

         feature_store_data["identifier"] = range(19)
         feature_store_data["mean_amount"] = 0.0
         feature_store_data["count"] = 1
         feature_store_data["EventTime"] = time.time()
```

Using the created dataframe we set the feature definitions

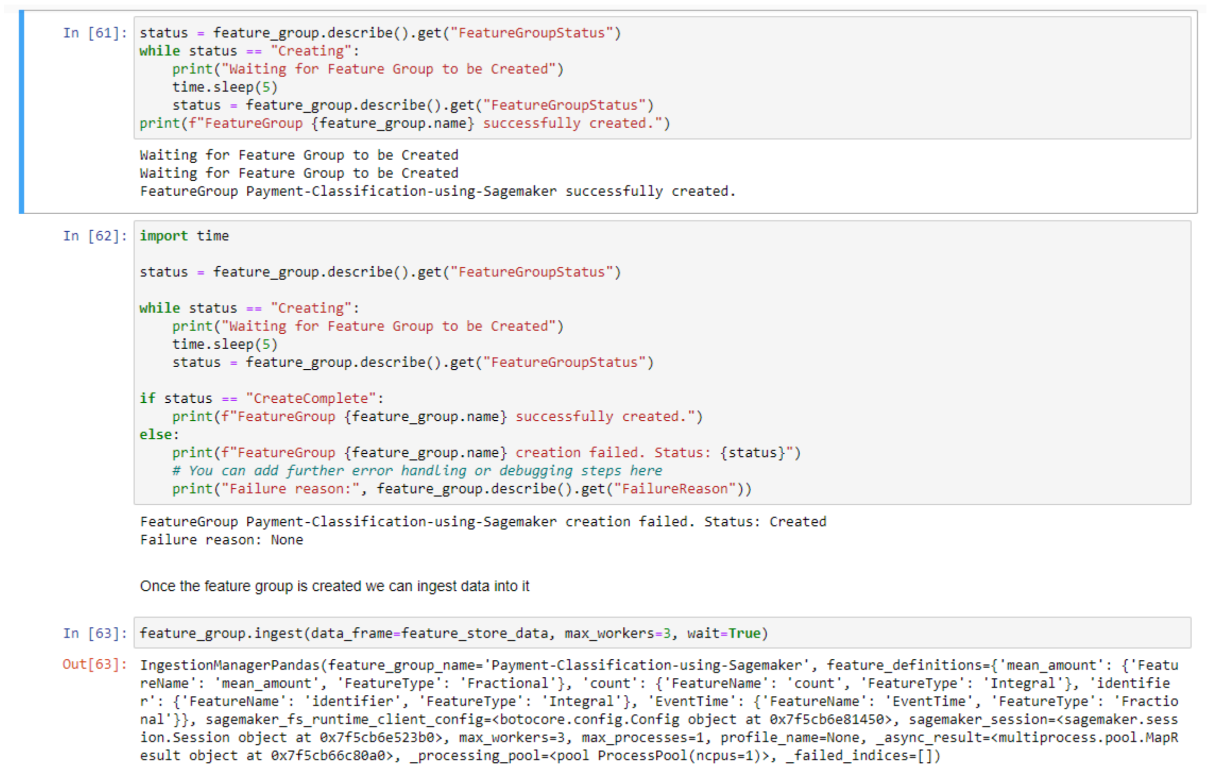
```
In [59]: feature_group.load_feature_definitions(data_frame=feature_store_data)

Out[59]: [FeatureDefinition(feature_name='mean_amount', feature_type=<FeatureTypeEnum.FRACTIONAL: 'Fractional'>, collection_type=None),
          FeatureDefinition(feature_name='count', feature_type=<FeatureTypeEnum.INTEGRAL: 'Integral'>, collection_type=None),
          FeatureDefinition(feature_name='identifier', feature_type=<FeatureTypeEnum.INTEGRAL: 'Integral'>, collection_type=None),
          FeatureDefinition(feature_name='EventTime', feature_type=<FeatureTypeEnum.FRACTIONAL: 'Fractional'>, collection_type=None)]
```

With these definitions ready we can create the feature group itself

```
In [60]: feature_group.create(
             s3_uri=f"s3://{s3_bucket}/{bucket_prefix}",
             record_identifier_name=record_identifier_feature_name,
             event_time_feature_name="EventTime",
             role_arn=role,
             enable_online_store=True,
         )

Out[60]: {'FeatureGroupArn': 'arn:aws:sagemaker:ap-south-1:872362024427:feature-group/Payment-Classification-using-Sagemaker',
          'ResponseMetadata': {'RequestId': '339a3764-90a7-4f64-99a5-575c52030fe0',
           'HTTPStatusCode': 200,
           'HTTPHeaders': {'x-amzn-requestid': '339a3764-90a7-4f64-99a5-575c52030fe0',
            'content-type': 'application/x-amz-json-1.1',
            'content-length': '116',
            'date': 'Sat, 20 Apr 2024 05:19:00 GMT'},
           'RetryAttempts': 0}}
```

It takes a couple of minutes for the feature group to be created, we need to wait for this to be done before trying to ingest data in the feature store

Assigning various Roles and Services in IAM user allows to get Database or Create tables in it and store the data. Checking the Feature Group Status allows us to understand that the S3 Buckets are created and Roles are assigned Successfully or not. Here in the S3 Bucket the Feature groups are assigned accordingly in the Servuices. If the User Roles and Services are successfully created then data is ingested into it by the various batches or Mini Batches.



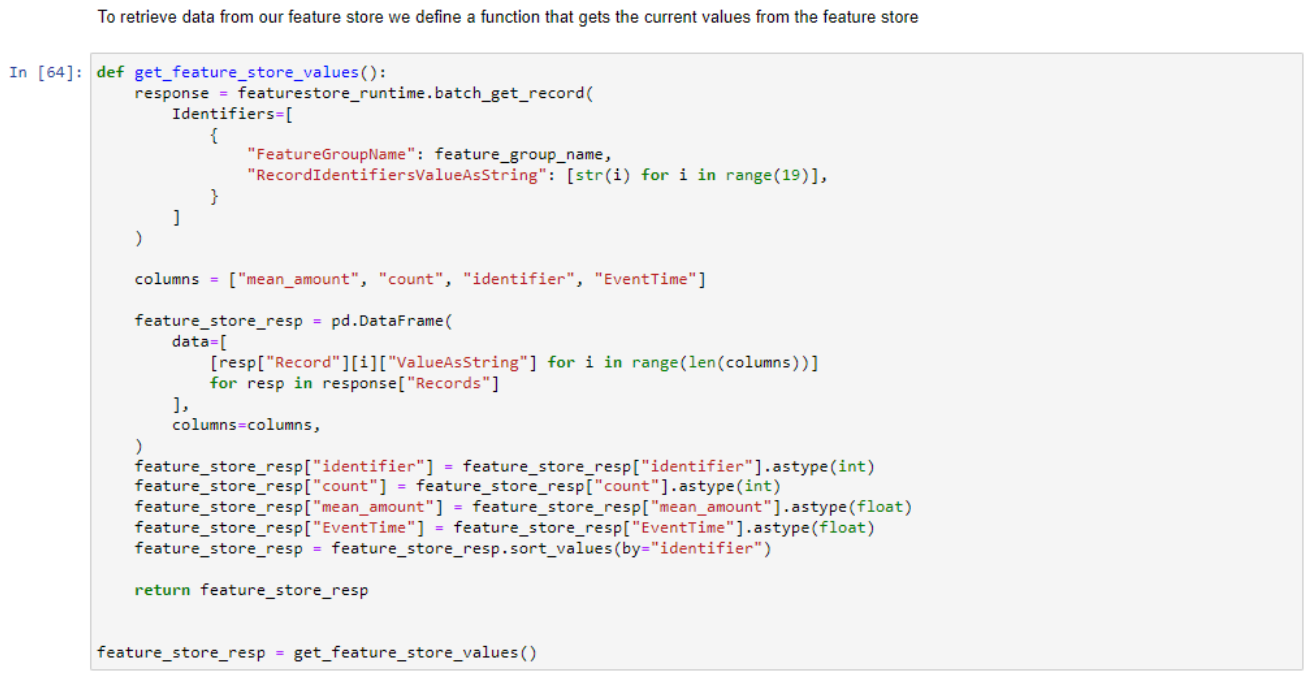
```
In [61]: status = feature_group.describe().get("FeatureGroupStatus")
         while status == "Creating":
             print("Waiting for Feature Group to be Created")
             time.sleep(5)
             status = feature_group.describe().get("FeatureGroupStatus")
         print(f"FeatureGroup {feature_group.name} successfully created.")

         Waiting for Feature Group to be Created
         Waiting for Feature Group to be Created
         FeatureGroup Payment-Classification-using-Sagemaker successfully created.
```

```
In [62]: import time

         status = feature_group.describe().get("FeatureGroupStatus")

         while status == "Creating":
             print("Waiting for Feature Group to be Created")
             time.sleep(5)
             status = feature_group.describe().get("FeatureGroupStatus")

         if status == "CreateComplete":
             print(f"FeatureGroup {feature_group.name} successfully created.")
         else:
             print(f"FeatureGroup {feature_group.name} creation failed. Status: {status}")
             # You can add further error handling or debugging steps here
             print("Failure reason:", feature_group.describe().get("FailureReason"))

         FeatureGroup Payment-Classification-using-Sagemaker creation failed. Status: Created
         Failure reason: None
```

Once the feature group is created we can ingest data into it

```
In [63]: feature_group.ingest(data_frame=feature_store_data, max_workers=3, wait=True)

Out[63]: IngestionManagerPandas(feature_group_name='Payment-Classification-using-Sagemaker', feature_definitions={'mean_amount': {'Featu
         reName': 'mean_amount', 'FeatureType': 'Fractional'}, 'count': {'FeatureName': 'count', 'FeatureType': 'Integral'}, 'identifie
         r': {'FeatureName': 'identifier', 'FeatureType': 'Integral'}, 'EventTime': {'FeatureName': 'EventTime', 'FeatureType': 'Fractio
         nal'}}, sagemaker_fs_runtime_client_config=<botocore.config.Config object at 0x7f5cb6e81450>, sagemaker_session=<sagemaker.sess
         ion.Session object at 0x7f5cb6e523b0>, max_workers=3, max_processes=1, profile_name=None, _async_result=<multiprocess.pool.MapR
         esult object at 0x7f5cb66c80a0>, _processing_pool=<pool ProcessPool(ncpus=1)>, _failed_indices=[])
```

We Retreive the data from the Feature Groups by defining a Function and getting the data from and storing it.



To retrieve data from our feature store we define a function that gets the current values from the feature store

```
In [64]: def get_feature_store_values():
             response = featurestore_runtime.batch_get_record(
                 Identifiers=[
                     {
                         "FeatureGroupName": feature_group_name,
                         "RecordIdentifiersValueAsString": [str(i) for i in range(19)],
                     }
                 ]
             )

             columns = ["mean_amount", "count", "identifier", "EventTime"]

             feature_store_resp = pd.DataFrame(
                 data=[
                     [resp["Record"][i]["ValueAsString"] for i in range(len(columns))]
                     for resp in response["Records"]
                 ],
                 columns=columns,
             )
             feature_store_resp["identifier"] = feature_store_resp["identifier"].astype(int)
             feature_store_resp["count"] = feature_store_resp["count"].astype(int)
             feature_store_resp["mean_amount"] = feature_store_resp["mean_amount"].astype(float)
             feature_store_resp["EventTime"] = feature_store_resp["EventTime"].astype(float)
             feature_store_resp = feature_store_resp.sort_values(by="identifier")

             return feature_store_resp


         feature_store_resp = get_feature_store_values()
```

This Data can now be manipulated by the Real values of the data which is Ingested into batches or Mini-Batches.

```
In [65]: feature_store_data = pd.DataFrame()
         feature_store_data["mean_amount"] = data.groupby(["transaction_category"]).mean()["amount"]
         feature_store_data["count"] = data.groupby(["transaction_category"]).count()["amount"]
         feature_store_data["identifier"] = feature_store_data.index
         feature_store_data["EventTime"] = time.time()

         feature_store_data["mean_amount"] = (
             pd.concat([feature_store_resp, feature_store_data])
             .groupby("identifier")
             .apply(lambda x: np.average(x["mean_amount"], weights=x["count"]))
         )
         feature_store_data["count"] = (
             pd.concat([feature_store_resp, feature_store_data]).groupby("identifier").sum()["count"]
         )

         feature_group.ingest(data_frame=feature_store_data, max_workers=3, wait=True)
```

```
/tmp/ipykernel_8736/1541770623.py:10: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavio
r is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `inclu
de_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
  .apply(lambda x: np.average(x["mean_amount"], weights=x["count"]))
```

```
Out[65]: IngestionManagerPandas(feature_group_name='Payment-Classification-using-Sagemaker', feature_definitions={'mean_amount': {'Featu
reName': 'mean_amount', 'FeatureType': 'Fractional'}, 'count': {'FeatureName': 'count', 'FeatureType': 'Integral'}, 'identifie
r': {'FeatureName': 'identifier', 'FeatureType': 'Integral'}, 'EventTime': {'FeatureName': 'EventTime', 'FeatureType': 'Fractio
nal'}}, sagemaker_fs_runtime_client_config=<botocore.config.Config object at 0x7f5cb6e81450>, sagemaker_session=<sagemaker.sess
ion.Session object at 0x7f5cb6e523b0>, max_workers=3, max_processes=1, profile_name=None, _async_result=<multiprocess.pool.MapR
esult object at 0x7f5cb7c5f640>, _processing_pool=<pool ProcessPool(ncpus=1)>, _failed_indices=[])
```

A display of features Store Data according to the real time Ingested data using Feature Group calling

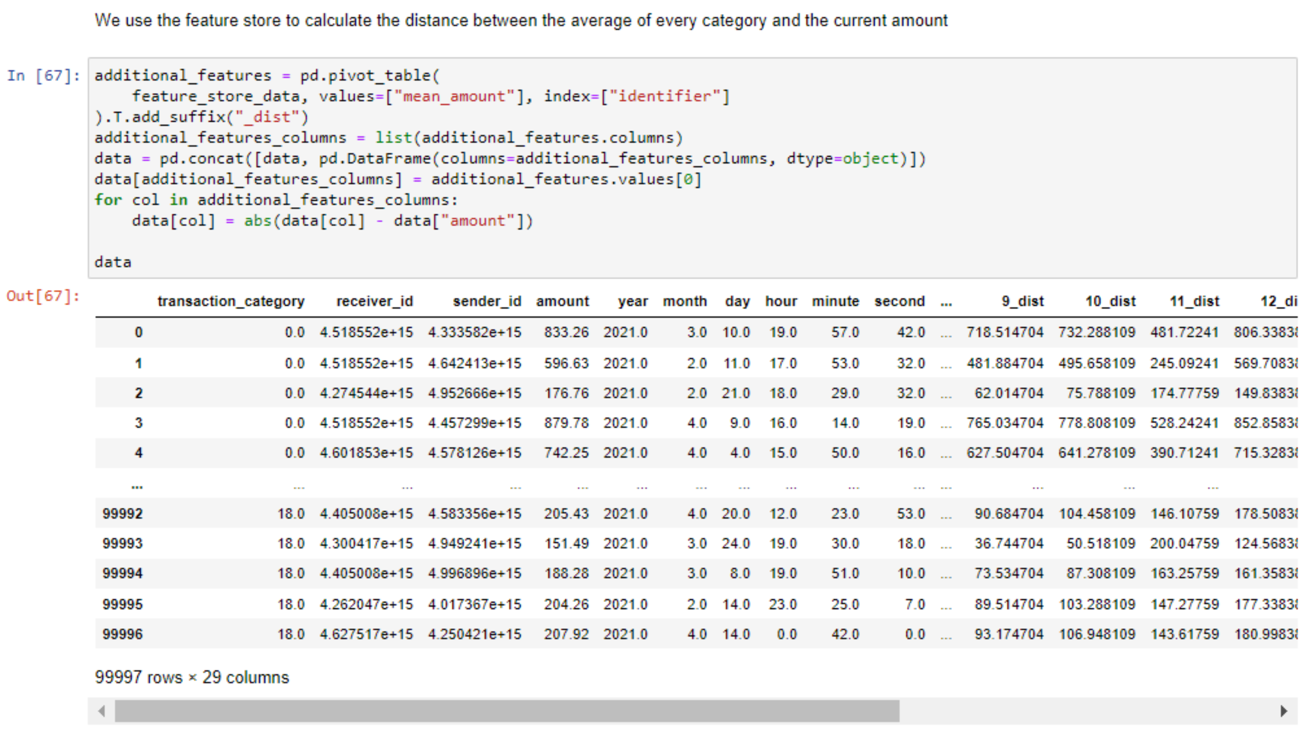
And display them after getting them from the feature store

```
In [66]: feature_store_data = get_feature_store_values()
         feature_store_data
```

Out[66]:

	mean_amount	count	identifier	EventTime
2	494.773326	466	0	1.713590e+09
15	51.205853	14513	1	1.713590e+09
11	850.011007	745	2	1.713590e+09
14	100.914531	33954	3	1.713590e+09
5	31.478000	1210	4	1.713590e+09
8	119.571685	4838	5	1.713590e+09
4	93.239755	9675	6	1.713590e+09
16	51.054155	2792	7	1.713590e+09
17	6018.076434	931	8	1.713590e+09
10	114.745296	3350	9	1.713590e+09
7	100.971891	19350	10	1.713590e+09
9	351.537590	1303	11	1.713590e+09
0	26.921611	931	12	1.713590e+09
3	204.592536	1396	13	1.713590e+09
18	496.232503	931	14	1.713590e+09
12	2907.420730	466	15	1.713590e+09
13	373.222299	187	16	1.713590e+09
6	780.972321	1861	17	1.713590e+09
1	205.273214	1117	18	1.713590e+09

Basic Manipulation is applied on the Columns



We use the feature store to calculate the distance between the average of every category and the current amount

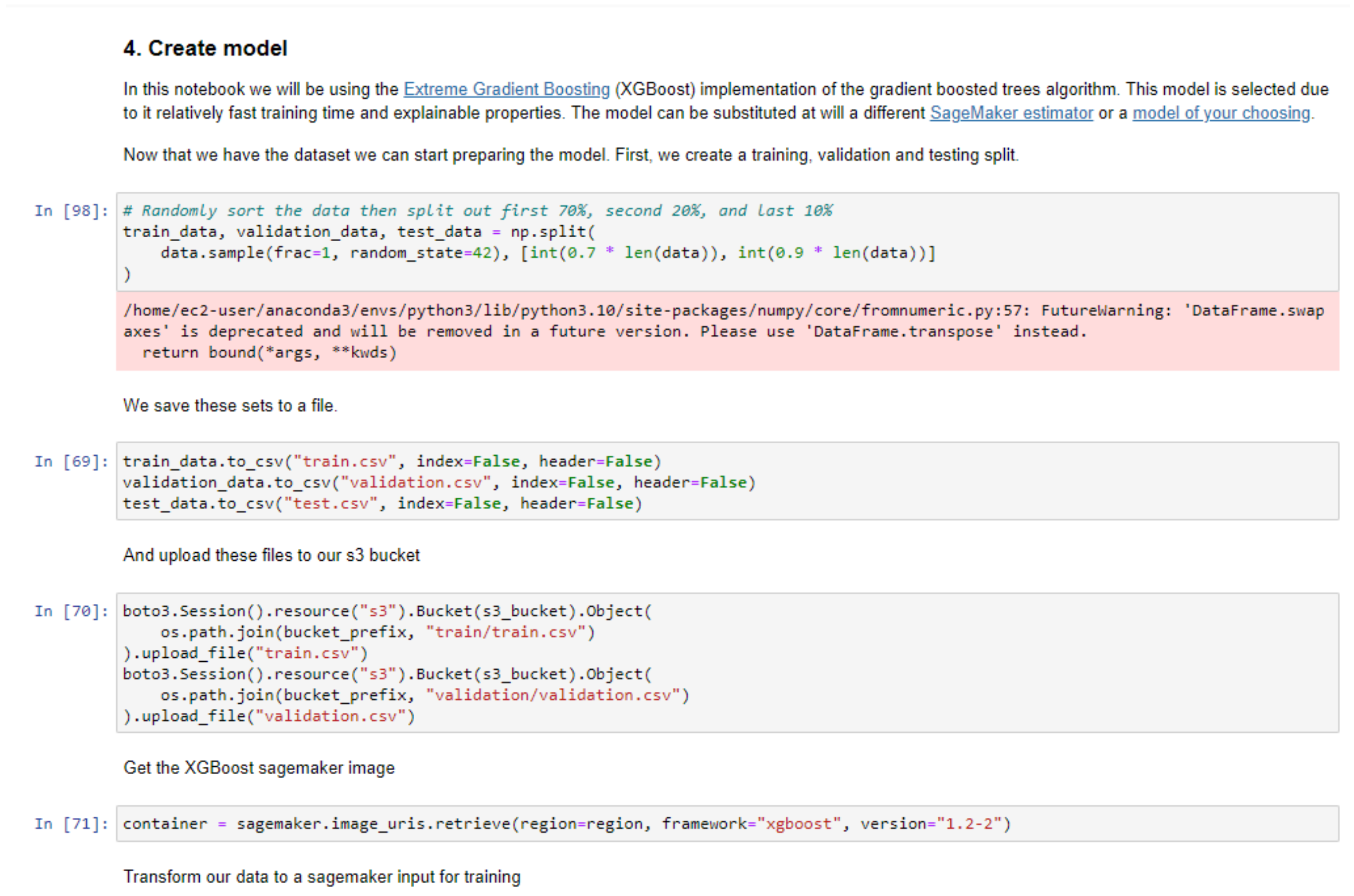
In [67]:
```python
additional_features = pd.pivot_table(
    feature_store_data, values=["mean_amount"], index=["identifier"]
).T.add_suffix("_dist")
additional_features_columns = list(additional_features.columns)
data = pd.concat([data, pd.DataFrame(columns=additional_features_columns, dtype=object)])
data[additional_features_columns] = additional_features.values[0]
for col in additional_features_columns:
    data[col] = abs(data[col] - data["amount"])

data
```

Out[67]:

	transaction_category	receiver_id	sender_id	amount	year	month	day	hour	minute	second	...	9_dist	10_dist	11_dist	12_di
0	0.0	4.518552e+15	4.333582e+15	833.26	2021.0	3.0	10.0	19.0	57.0	42.0	...	718.514704	732.288109	481.72241	806.3383
1	0.0	4.518552e+15	4.642413e+15	596.63	2021.0	2.0	11.0	17.0	53.0	32.0	...	481.884704	495.658109	245.09241	569.7083
2	0.0	4.274544e+15	4.952666e+15	176.76	2021.0	2.0	21.0	18.0	29.0	32.0	...	62.014704	75.788109	174.77759	149.8383
3	0.0	4.518552e+15	4.457299e+15	879.78	2021.0	4.0	9.0	16.0	14.0	19.0	...	765.034704	778.808109	528.24241	852.8583
4	0.0	4.601853e+15	4.578126e+15	742.25	2021.0	4.0	4.0	15.0	50.0	16.0	...	627.504704	641.278109	390.71241	715.3283
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
99992	18.0	4.405008e+15	4.583356e+15	205.43	2021.0	4.0	20.0	12.0	23.0	53.0	...	90.684704	104.458109	146.10759	178.5083
99993	18.0	4.300417e+15	4.949241e+15	151.49	2021.0	3.0	24.0	19.0	30.0	18.0	...	36.744704	50.518109	200.04759	124.5683
99994	18.0	4.405008e+15	4.996896e+15	188.28	2021.0	3.0	8.0	19.0	51.0	10.0	...	73.534704	87.308109	163.25759	161.3583
99995	18.0	4.262047e+15	4.017367e+15	204.26	2021.0	2.0	14.0	23.0	25.0	7.0	...	89.514704	103.288109	147.27759	177.3383
99996	18.0	4.627517e+15	4.250421e+15	207.92	2021.0	4.0	14.0	0.0	42.0	0.0	...	93.174704	106.948109	143.61759	180.9983

99997 rows × 29 columns

A Model is used in the Amazon Sagemaker and now it can be used for Faster Training and Inferences. Here the data is Splitted into Training, Testing and validation data Now the various Files will be uploaded to S3 Bucket. Various Machine models will have their own Specific Image in Sagemaker and only those Images are used to train the Model on the training Dataset



4. Create model

In this notebook we will be using the Extreme Gradient Boosting (XGBoost) implementation of the gradient boosted trees algorithm. This model is selected due to it relatively fast training time and explainable properties. The model can be substituted at will a different SageMaker estimator or a model of your choosing.

Now that we have the dataset we can start preparing the model. First, we create a training, validation and testing split.

In [98]:
```
# Randomly sort the data then split out first 70%, second 20%, and last 10%
train_data, validation_data, test_data = np.split(
    data.sample(frac=1, random_state=42), [int(0.7 * len(data)), int(0.9 * len(data))]
)
```
```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/numpy/core/fromnumeric.py:57: FutureWarning: 'DataFrame.swap
axes' is deprecated and will be removed in a future version. Please use 'DataFrame.transpose' instead.
  return bound(*args, **kwds)
```

We save these sets to a file.

In [69]:
```
train_data.to_csv("train.csv", index=False, header=False)
validation_data.to_csv("validation.csv", index=False, header=False)
test_data.to_csv("test.csv", index=False, header=False)
```

And upload these files to our s3 bucket

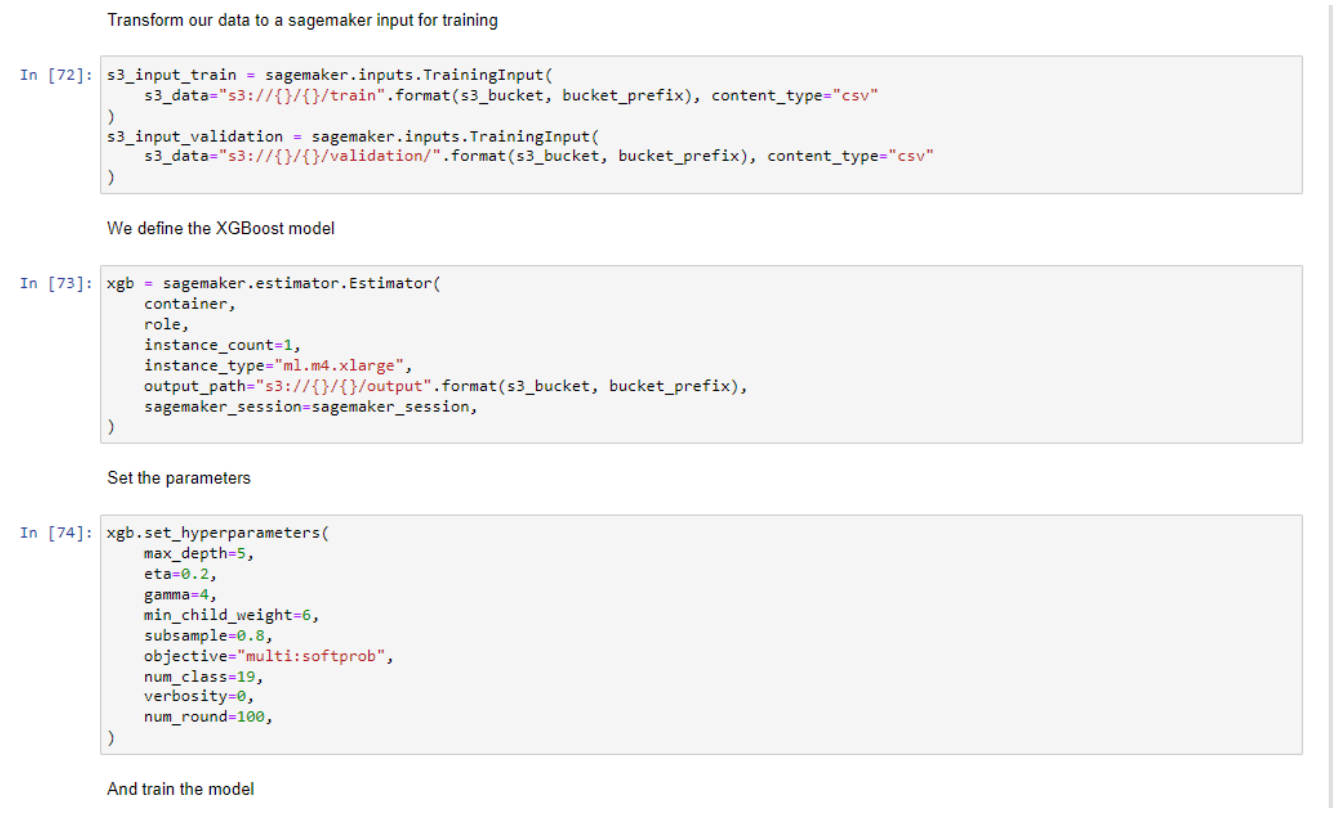
In [70]:
```
boto3.Session().resource("s3").Bucket(s3_bucket).Object(
    os.path.join(bucket_prefix, "train/train.csv")
).upload_file("train.csv")
boto3.Session().resource("s3").Bucket(s3_bucket).Object(
    os.path.join(bucket_prefix, "validation/validation.csv")
).upload_file("validation.csv")
```

Get the XGBoost sagemaker image

In [71]:
```
container = sagemaker.image_uris.retrieve(region=region, framework="xgboost", version="1.2-2")
```

Transform our data to a sagemaker input for training

Data is Further transformed to pass on as input to Model for Training and now this we define into the XGBoost Model and set the parameters (Hyperparameter tuning)



Transform our data to a sagemaker input for training

In [72]:
```
s3_input_train = sagemaker.inputs.TrainingInput(
    s3_data="s3://{}/{}/train".format(s3_bucket, bucket_prefix), content_type="csv"
)
s3_input_validation = sagemaker.inputs.TrainingInput(
    s3_data="s3://{}/{}/validation/".format(s3_bucket, bucket_prefix), content_type="csv"
)
```

We define the XGBoost model

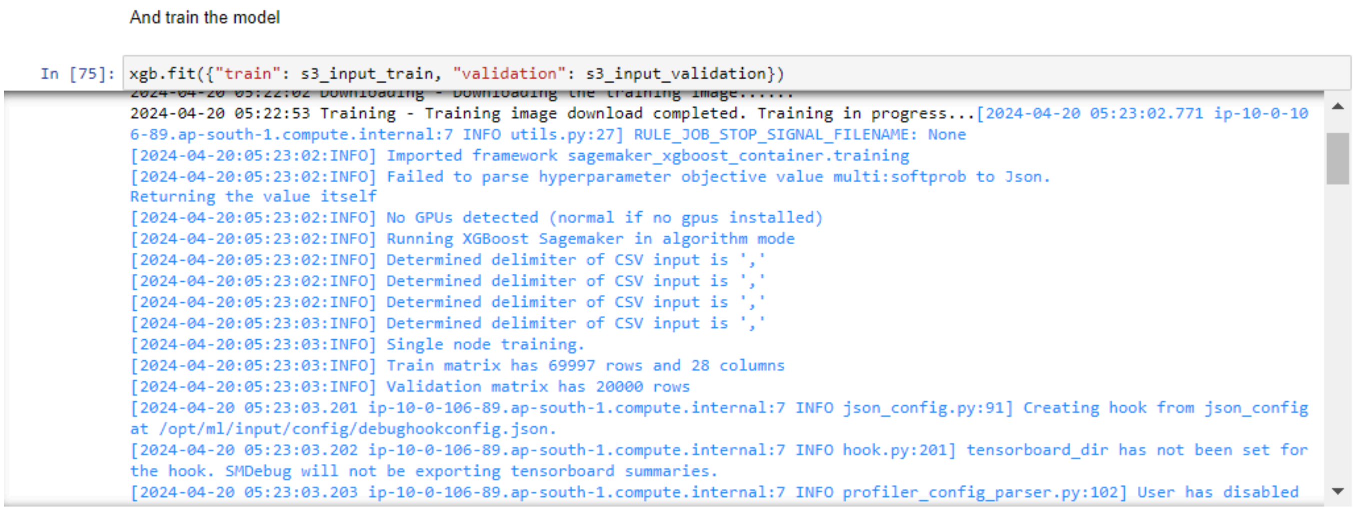
In [73]:
```
xgb = sagemaker.estimator.Estimator(
    container,
    role,
    instance_count=1,
    instance_type="ml.m4.xlarge",
    output_path="s3://{}/{}/output".format(s3_bucket, bucket_prefix),
    sagemaker_session=sagemaker_session,
)
```

Set the parameters

In [74]:
```
xgb.set_hyperparameters(
    max_depth=5,
    eta=0.2,
    gamma=4,
    min_child_weight=6,
    subsample=0.8,
    objective="multi:softprob",
    num_class=19,
    verbosity=0,
    num_round=100,
)
```

And train the model

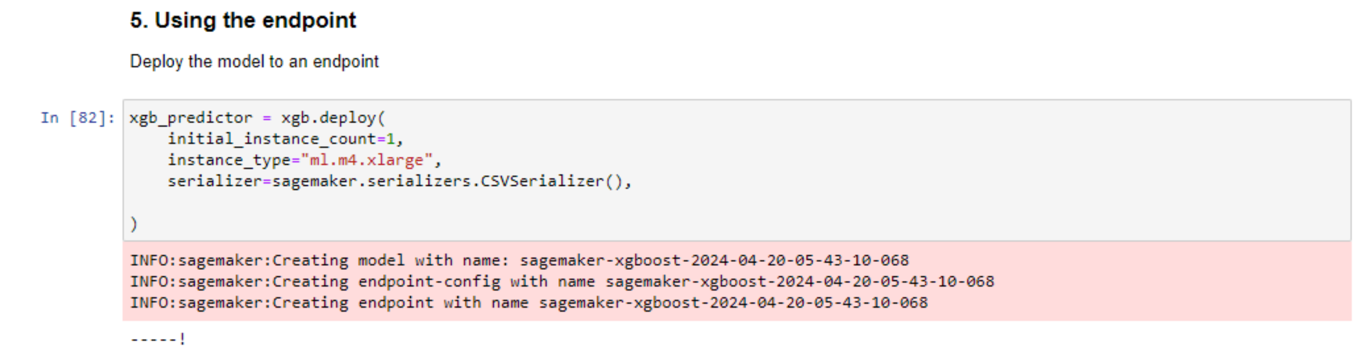
Further training the Model and storing its endpoints so that by using these Endpoints the Sagemaker can call the AWS Endpoint through various to predict for Payment Classification



And train the model

```
In [75]: xgb.fit({"train": s3_input_train, "validation": s3_input_validation})
```

```
2024-04-20 05:22:02 Downloading - Downloading the training image......
2024-04-20 05:22:53 Training - Training image download completed. Training in progress...[2024-04-20 05:23:02.771 ip-10-0-10
6-89.ap-south-1.compute.internal:7 INFO utils.py:27] RULE_JOB_STOP_SIGNAL_FILENAME: None
[2024-04-20:05:23:02:INFO] Imported framework sagemaker_xgboost_container.training
[2024-04-20:05:23:02:INFO] Failed to parse hyperparameter objective value multi:softprob to Json.
Returning the value itself
[2024-04-20:05:23:02:INFO] No GPUs detected (normal if no gpus installed)
[2024-04-20:05:23:02:INFO] Running XGBoost Sagemaker in algorithm mode
[2024-04-20:05:23:02:INFO] Determined delimiter of CSV input is ','
[2024-04-20:05:23:02:INFO] Determined delimiter of CSV input is ','
[2024-04-20:05:23:02:INFO] Determined delimiter of CSV input is ','
[2024-04-20:05:23:03:INFO] Determined delimiter of CSV input is ','
[2024-04-20:05:23:03:INFO] Single node training.
[2024-04-20:05:23:03:INFO] Train matrix has 69997 rows and 28 columns
[2024-04-20:05:23:03:INFO] Validation matrix has 20000 rows
[2024-04-20 05:23:03.201 ip-10-0-106-89.ap-south-1.compute.internal:7 INFO json_config.py:91] Creating hook from json_config
at /opt/ml/input/config/debughookconfig.json.
[2024-04-20 05:23:03.202 ip-10-0-106-89.ap-south-1.compute.internal:7 INFO hook.py:201] tensorboard_dir has not been set for
the hook. SMDebug will not be exporting tensorboard summaries.
[2024-04-20 05:23:03.203 ip-10-0-106-89.ap-south-1.compute.internal:7 INFO profiler_config_parser.py:102] User has disabled
```



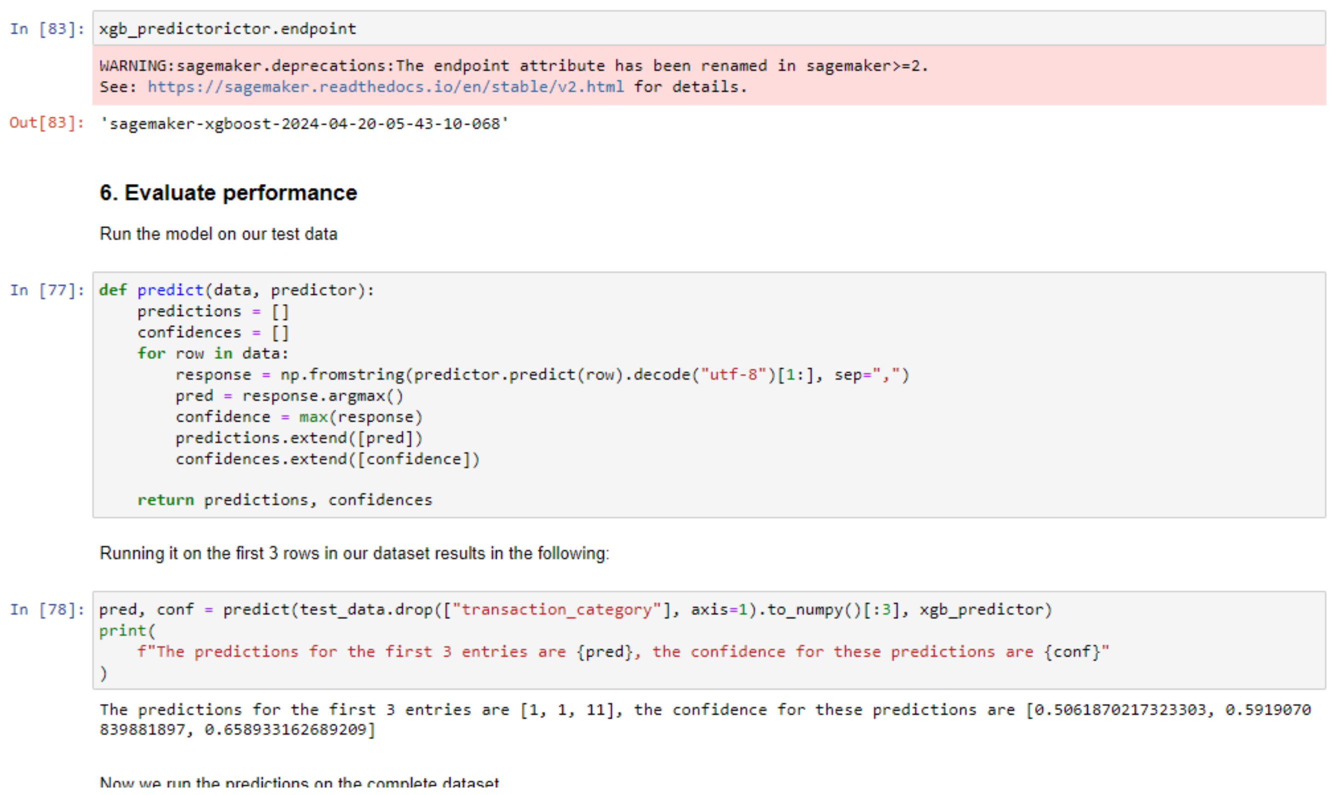
5. Using the endpoint

Deploy the model to an endpoint

```
In [82]: xgb_predictor = xgb.deploy(
             initial_instance_count=1,
             instance_type="ml.m4.xlarge",
             serializer=sagemaker.serializers.CSVSerializer(),

         )
```

```
INFO:sagemaker:Creating model with name: sagemaker-xgboost-2024-04-20-05-43-10-068
INFO:sagemaker:Creating endpoint-config with name sagemaker-xgboost-2024-04-20-05-43-10-068
INFO:sagemaker:Creating endpoint with name sagemaker-xgboost-2024-04-20-05-43-10-068
-----!
```

An Evaluation Metrics are defined here for the Model to further enhance and improve the capabilities



```
In [83]: xgb_predictorictor.endpoint
```

```
WARNING:sagemaker.deprecations:The endpoint attribute has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
```

```
Out[83]: 'sagemaker-xgboost-2024-04-20-05-43-10-068'
```

6. Evaluate performance

Run the model on our test data

```
In [77]: def predict(data, predictor):
             predictions = []
             confidences = []
             for row in data:
                 response = np.fromstring(predictor.predict(row).decode("utf-8")[1:], sep=",")
                 pred = response.argmax()
                 confidence = max(response)
                 predictions.extend([pred])
                 confidences.extend([confidence])

             return predictions, confidences
```

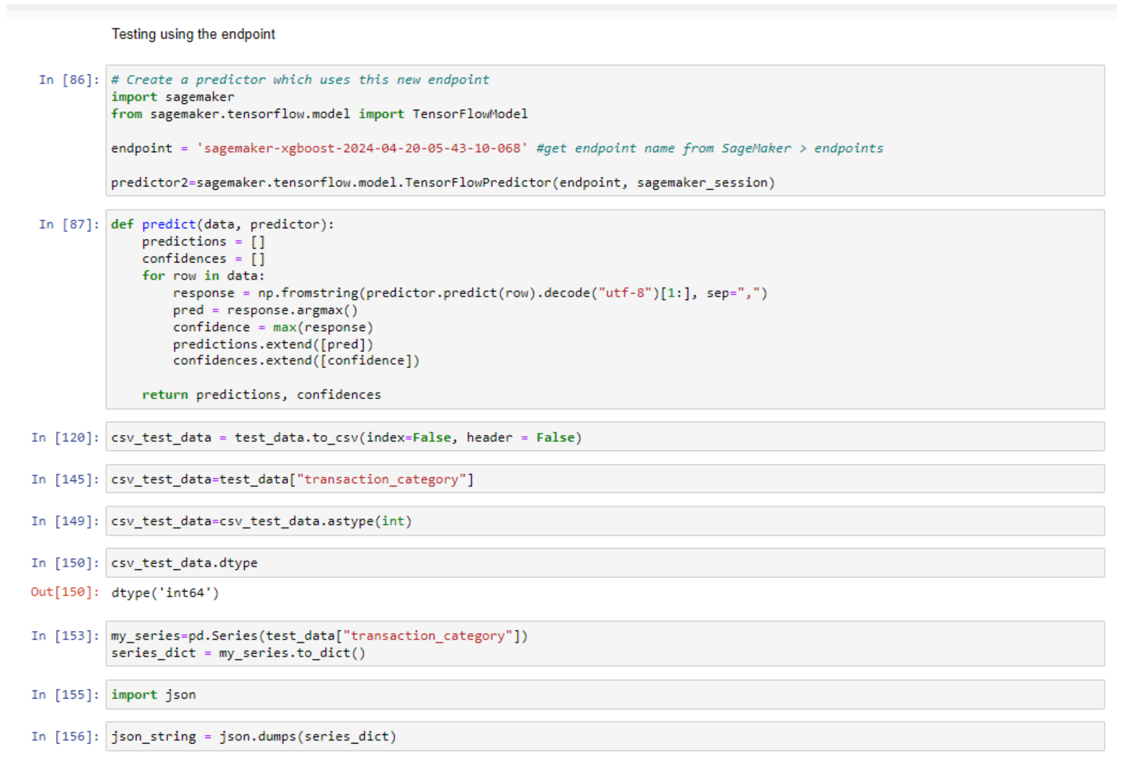
Running it on the first 3 rows in our dataset results in the following:

```
In [78]: pred, conf = predict(test_data.drop(["transaction_category"], axis=1).to_numpy()[:3], xgb_predictor)
         print(
             f"The predictions for the first 3 entries are {pred}, the confidence for these predictions are {conf}"
         )
```

```
The predictions for the first 3 entries are [1, 1, 11], the confidence for these predictions are [0.5061870217323303, 0.5919070
839881897, 0.658933162689209]
```

Now we run the predictions on the complete dataset

Further in the last part we test the endpoint by passing the test.csv file but the model does not provide the correct output as there is an Client Error mentioned over there and since it could not be resolved by various Methods, it can be added as Future Scope and used for Multiple Endpoint Inferences



Testing using the endpoint

```
In [86]: # Create a predictor which uses this new endpoint
         import sagemaker
         from sagemaker.tensorflow.model import TensorFlowModel

         endpoint = 'sagemaker-xgboost-2024-04-20-05-43-10-068' #get endpoint name from SageMaker > endpoints

         predictor2=sagemaker.tensorflow.model.TensorFlowPredictor(endpoint, sagemaker_session)
```

```
In [87]: def predict(data, predictor):
             predictions = []
             confidences = []
             for row in data:
                 response = np.fromstring(predictor.predict(row).decode("utf-8")[1:], sep=",")
                 pred = response.argmax()
                 confidence = max(response)
                 predictions.extend([pred])
                 confidences.extend([confidence])

             return predictions, confidences
```

```
In [120]: csv_test_data = test_data.to_csv(index=False, header = False)
```

```
In [145]: csv_test_data=test_data["transaction_category"]
```

```
In [149]: csv_test_data=csv_test_data.astype(int)
```

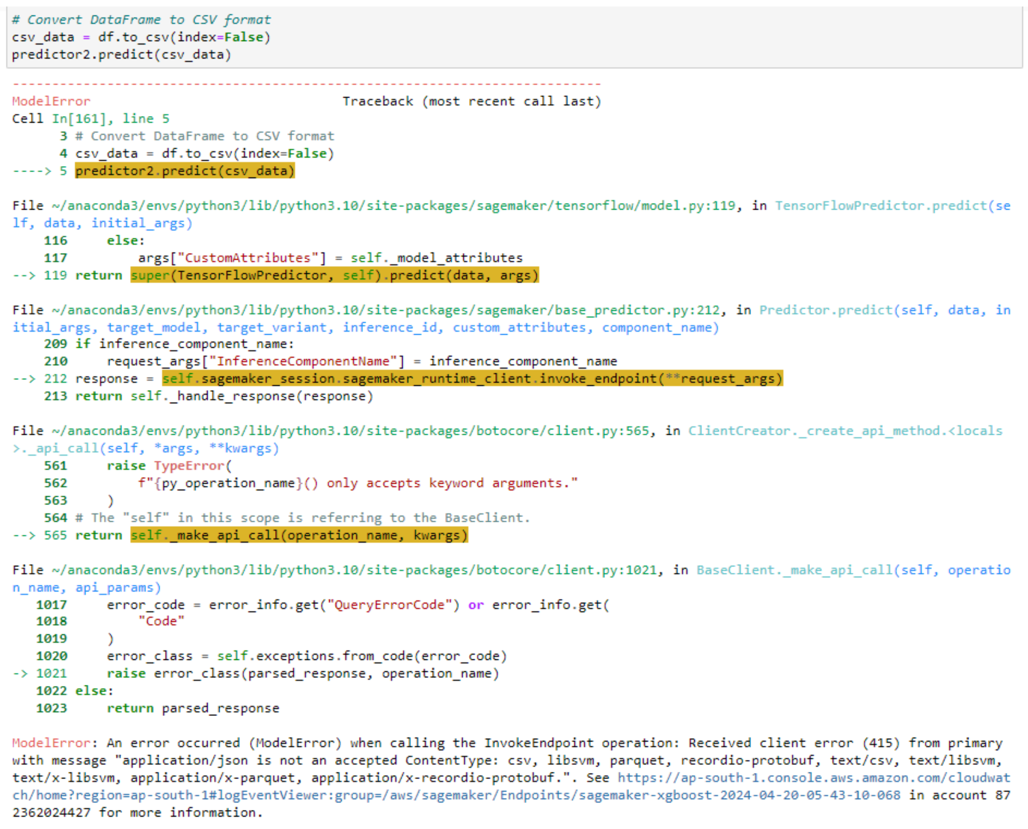
```
In [150]: csv_test_data.dtype
Out[150]: dtype('int64')
```

```
In [153]: my_series=pd.Series(test_data["transaction_category"])
          series_dict = my_series.to_dict()
```

```
In [155]: import json
```

```
In [156]: json_string = json.dumps(series_dict)
```

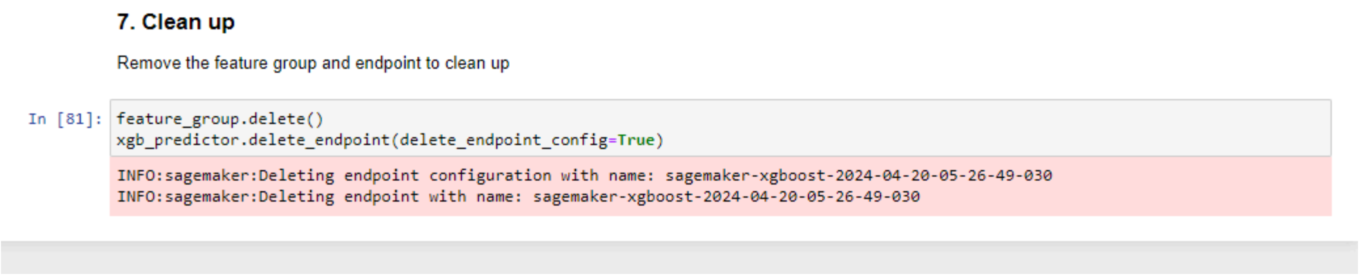
Here is the Error which was not able to resolve but the Endpoint is successfully created and Model is Deployed including the evaluation Metrics and Endpoint gets deleted too.



```
# Convert DataFrame to CSV format
csv_data = df.to_csv(index=False)
predictor2.predict(csv_data)
```

```
---------------------------------------------------------------------------
ModelError                                Traceback (most recent call last)
Cell In[161], line 5
      3 # Convert DataFrame to CSV format
      4 csv_data = df.to_csv(index=False)
----> 5 predictor2.predict(csv_data)

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/sagemaker/tensorflow/model.py:119, in TensorFlowPredictor.predict(se
lf, data, initial_args)
    116     else:
    117         args["CustomAttributes"] = self._model_attributes
--> 119 return super(TensorFlowPredictor, self).predict(data, args)

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/sagemaker/base_predictor.py:212, in Predictor.predict(self, data, in
itial_args, target_model, target_variant, inference_id, custom_attributes, component_name)
    209 if inference_component_name:
    210     request_args["InferenceComponentName"] = inference_component_name
--> 212 response = self.sagemaker_session.sagemaker_runtime_client.invoke_endpoint(**request_args)
    213 return self._handle_response(response)

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/botocore/client.py:565, in ClientCreator._create_api_method.<locals
>._api_call(self, *args, **kwargs)
    561     raise TypeError(
    562         f"{py_operation_name}() only accepts keyword arguments."
    563     )
    564 # The "self" in this scope is referring to the BaseClient.
--> 565 return self._make_api_call(operation_name, kwargs)

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/botocore/client.py:1021, in BaseClient._make_api_call(self, operatio
n_name, api_params)
   1017     error_code = error_info.get("QueryErrorCode") or error_info.get(
   1018         "Code"
   1019     )
   1020     error_class = self.exceptions.from_code(error_code)
-> 1021     raise error_class(parsed_response, operation_name)
   1022 else:
   1023     return parsed_response

ModelError: An error occurred (ModelError) when calling the InvokeEndpoint operation: Received client error (415) from primary
with message "application/json is not an accepted ContentType: csv, libsvm, parquet, recordio-protobuf, text/csv, text/libsvm,
text/x-libsvm, application/x-parquet, application/x-recordio-protobuf.". See https://ap-south-1.console.aws.amazon.com/cloudwat
ch/home?region=ap-south-1#logEventViewer:group=/aws/sagemaker/Endpoints/sagemaker-xgboost-2024-04-20-05-43-10-068 in account 87
2362024427 for more information.
```

Cleaning the data means removing the Endpoints from the Notebook/ model used for solving Problem Statement



7. Clean up

Remove the feature group and endpoint to clean up

```
In [81]: feature_group.delete()
         xgb_predictor.delete_endpoint(delete_endpoint_config=True)

INFO:sagemaker:Deleting endpoint configuration with name: sagemaker-xgboost-2024-04-20-05-26-49-030
INFO:sagemaker:Deleting endpoint with name: sagemaker-xgboost-2024-04-20-05-26-49-030
```

Future Scope

Now in this if we talk about the Resources which were used to deploy this Model the methods were very limited and this cannot be further used for larger Problem Statements. Due to this we require Amazon Kinesis to store the User Data and pass this to Amazon Sagemaker. Now further using Amazon Lambda Function we can transform this data to various different Methods and then use Amazon SageMaker and Amazon Security Lake(For Security and Autoomation Scaling purpose). This Method can be used as Template for Large Problem Statements and Machine Learning/Deep Learning Models to Deploy.

References

PwC and AXS Merger including AWS Implementation
https://www.pwc.com/us/en/library/case-studies/axs.html

Data and Analytics
https://www.pwc.com/us/en/library/case-studies/wyndham-data-architecture.html

Security Lake
https://www.pwc.com/us/en/technology/alliances/amazon-web-services/fusion-center.html

PWC Services
https://www.pwc.com/us/en/technology/alliances/amazon-web-services.html#:~:text=Explore%20the%20power%20of%20PwC,efficiency%20of%20your%20cloud%20 investments.

https://aws.amazon.com/blogs/apn/fusing-business-data-with-pwc-enterprise-data-platform-for-a-unified-data-strategy/#:~:text=PwC%20is%20an%20AWS%20Premier,compete%20in%20today's%20service%20economy.