**TECHNICAL ASSESSMENT**
**AI AND DATA ENGINEER ROLE – VAN LANSCHOT KEMPEN**
AUTHOR: ARYAN OMKAR ASHAR

## Disclaimer

The dataset as well as the presented results from the pipeline is based on synthetic data generation process which is completely random and created solely for educational and demonstration purposes as part of the technical assessment provided by Van Lanschot Kempen for the position of AI and Data Engineer. It does not contain any real or personally identifiable information (PII), and any resemblance to actual individuals, entities, or transactions is purely coincidental. Additionally, this dataset, by no means, represents any personal political stance of the owner. The dataset has been artificially generated to simulate typical KYC data structures, including client, transaction, and compliance information, to demonstrate data ingestion, transformation, risk scoring, and machine learning techniques in a controlled environment.

This dataset:

- Should not be used for production or regulatory purposes.
- Does not represent real financial activity or risk.
- Is compliant with data privacy and confidentiality standards, as no sensitive data has been used.

All outputs, analyses, and visualizations derived from this dataset are for illustrative purposes only.

## Execution of the code

The code can be executed and the results can be obtained by executing the following steps:

1. Installing all the libraries listed in "*requirements.txt"*
2. Delete all the output folders if you have installed them from GitHub
    a. *data*
    b. *final_output_dataset*
    c. *models*
3. Running the python script titled *"main_pipeline.py"*

## Explanation of the Dataset

The entirety of the dataset has been generates randomly and is a form of synthetic dataset. The following is the structure of the dataset which has been generated. All the CSV files of the raw dataset (dataset at the start of the pipeline without any joining, enrichment and results) can be found in the folder titled *"raw data from data ingestion"*

1. *Clients.csv*
   This dataset is solely about the information collected from clients. The columns denote:

| Column Name | Description |
|---|---|
| client_id | The unique ID number of the client |
| name | Name of the client |
| date_of_birth | The DOB of client in YYYY-MM-DD |
| age | The age of the client |
| country | Country of citizenship of the client |
| email | Email address of the client |
| registration_date | Date on which the client registered with the financial institution |
| kyc_status | Verified, Under Review or Pending |
| customer_type | Individual, Business, Corporate |

2. *documents.csv*

This dataset contains information about the documents which have been submitted by the clients.

| **Column Name** | **Description** |
|---|---|
| document_id | The unique ID of the document submitted. |
| client_id | The ID of the client submitting the document. |
| document_type | Utility Bill, Bank Statement, Passport, ID Card, |
| verification_status | Verified, Failed, Pending |
| upload_date | The date of upload of the document in YYYY – MM – DD |

3. *countries.csv*

This dataset contains a list of 10 selected countries and their level of risk associated with each country.

| **Column Name** | **Description** |
|---|---|
| country | The 10 countries chosen for this assignment. |
| risk_level | The risk level associated with the country (Low, Medium and High) |
| aml_compliance_score | Numerical rating to assess involvement in financial crime |

4. *transactions.csv*

This dataset consists of all the transactions that have been carried out by the clients.

| **Column Name** | **Description** |
|---|---|
| transaction_id | The unique ID number of every transaction |
| client_id | The ID number of the client initiating the transaction |
| transaction_amount | The amount initiated for transaction |
| transaction_date | The date of transaction in last 3 months |
| transaction_type | Deposit, withdrawal, transfer or Investment |
| status | Completed, Pending or Failed |
| origin_country | Country of citizenship of the client. |
| destination_county | Country where the money is going to. |

5. *compliance_rules.csv*

This dataset consists of the rules which need to be followed by the clients and the transactions.

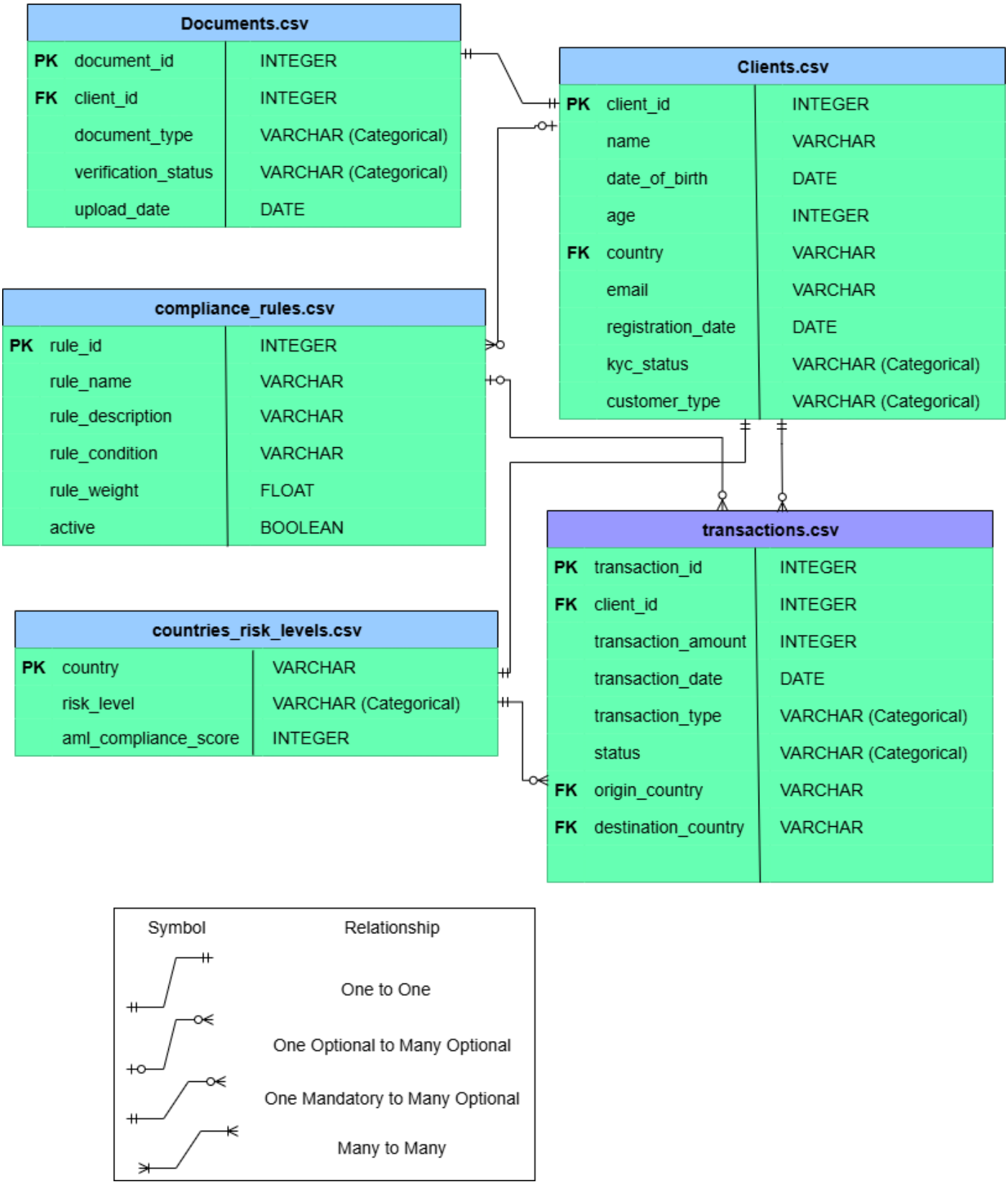
| **Column Name** | **Description** |
|---|---|
| rule_id | The unique ID number of every rule |
| rule_name | Name of the rule also indicating whether it is for the clients or for transactions |
| rule_description | Detailed Description of the rule |
| rule_condition | Computational condition of the rule |
| risk_weight | The risk score associated with the non-compliance of the rule. |
| active | A Boolean which controls whether the rule is active or not. |

**Task 2.2: - Data Ingestion**

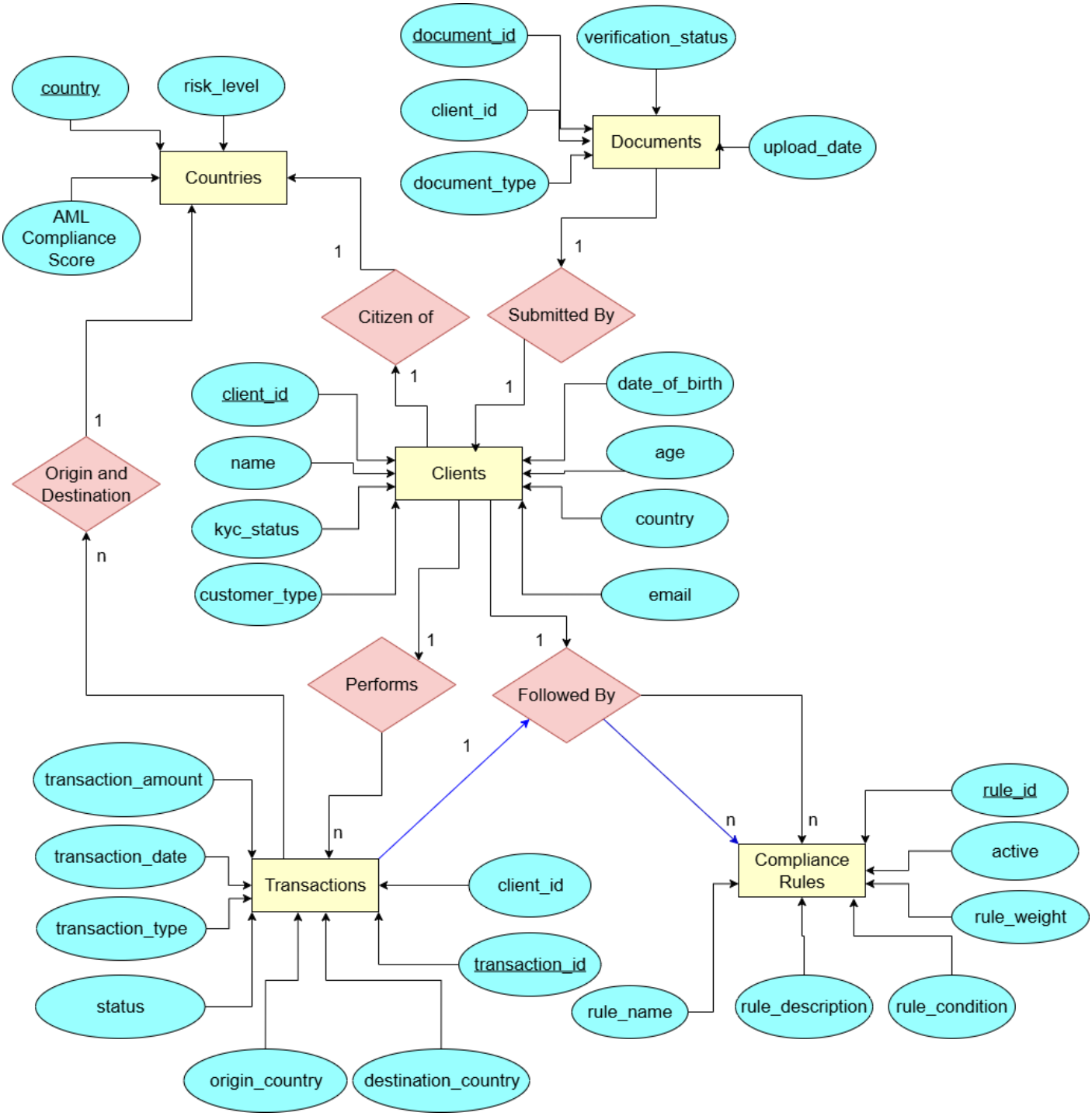
The code for this task can be found in the folder named *"data_ingestion"* and the file name of the python script is *"ingest_generate_data.py"*. As stated in the task, this task consists of generating the dataset and the dataset has been extended to 3 new datasets as allowed in the assignment. Additionally, at the end of the script, the dataframes are loaded from the CSV files which have been generated.

**Task 2.3: - Data Modeling**

This task consists of designing a scalable and flexible data model for the given assignment. The ER diagrams in 2 different notations (explaining the same thing) have been depicted below. These ER diagrams show the data model (made using draw io).




**Entity Relationship Diagram of the Data Model**




**Entity-Relationship Diagram of the Data Model with the Relationship**

This data model is both scalable and flexible. This data model is scalable since it supports high-volume transactional data through partitioning and distributed storage. New clients, transactions, or documents can be added without impacting existing tables. The clients table can be scaled horizontally by sharding by region or client type and the transactions table can be partitioned by transaction_date or client_id. Also Databricks delta lake has been used which supports incremental updates and makes the model scalable and data grows.

The proposed data model is flexible as well since all the compliance rules are saved in a separate folder with a flag controlling if the rule has to be kept active or not and these are not hard-coded in the pipeline. New rules can be added and old ones can be removed from the file. Linking is done via primary keys making it easy to extend the model to new unstructured sources. Additionally, more attributes can be added to each entity. The tables and entities are decoupled and modular. Adding a new data source or entity (e.g., accounts, loans, chat logs) does not require modifying existing tables. You just add a new entity and link via a foreign key.

**Task 2.4 Data Processing**

### 1. Join and Enrichment

This part focuses on joining and deriving the various features which will be useful of applying the business rules as well as getting results from the machine learning pipeline and models. The entire code for this part can be found in the folder "*data_joining_enrichment"* and the file is titles "*enrichment_pipeline.py*"

The data generated by running this code is saved in the folder *"data/joined and enriched data".* A total of 3 files are formed by the help of this script cited above.

*a. enriched_clients.csv*

| Column Header | Description | Derived From |
|---|---|---|
| client_id | The unique ID number of the client | Clients.csv |
| Name | Name of the client | |
| date_of_birth | The DOB of client in YYYY-MM-DD | |
| age | The age of the client | |
| country | Country of citizenship of the client | |
| Email | Email address of the client | |
| registration_date | Date on which the client registered with the financial institution | |
| kyc_status | Verified, Under Review or Pending | |
| customer_type | Individual, Business, Corporate | |
| total_transactions | Total transactions made by the client | Transactions.csv |
| total_amount | Total amount of all the transactions | |
| avg_amount, | Avg. amount of all transactions | |
| max_amount | Max. amount of all transactions | |
| min_amount | Min. amount of all transactions | |
| amount_std | Std. Deviation of all amount of all transactions | |
| failed_transaction_count | Number of failed transactions | |
| cross_border_count | Number of transactions sent across the border | countries.csv |
| high_risk_cross_border_count | Number of transactions sent across the border to high risk country | |
| country_orig | Country of origin of client | |
| client_country_risk_level | Risk level of the citizenship country of the client | |
| client_country_aml_score | AML score of the citizenship country of the client | |
| doc_verification_status | Verification status of the documents of the client | Documents.csv |
| tx_last_7 | Number of transactions in last 7 days | Transactions.csv |
| tx_last_30 | Number of transactions in last 30 days | |
| large_tx_last_7_days | Number of transactions with amount>10,000 euros in last 7 days | |

*b. enriched_transactions.csv*

| Column Header | Description | Derived From |
|---|---|---|
| transaction_id | The unique ID number of every transaction | Transactions.csv |
| client_id | The ID number of the client initiating the transaction | |
| transaction_amount | The amount initiated for transaction | |
| transaction_date | The date of transaction in last 3months | |
| transaction_type | Deposit, withdrawal, transfer or Investment | |
| status | Completed, Pending or Failed | |
| origin_country | Country of citizenship of the client. | |
| destination_county | Country where the money is going to. | |
| Name | Name of the client | Clients.csv |
| age | The age of the client | |
| country | Country of citizenship of the client | |
| kyc_status | Verified, Under Review or Pending of the client | |
| customer_type | Individual, Business, Corporate | |
| registration_date | Date of Registration of the client who made the transaction | |
| *origin_risk_level* | Risk level of the transaction's origin country | countries.csv |
| *origin_aml_score* | AML score of the transaction's origin country | |
| *destination_risk_level* | Risk level of the transaction's destination country | |
| *destination_aml_score* | AML score of the transaction's destination country | |
| *document_type* | Type of document submitted for KYC | Documents.csv |
| *doc_verification_status* | Status of verification of the document | |
| *doc_upload_date* | Upload date of the document | |
| *days_since_transaction* | Number of days since the transaction | Derived Feature |
| *days_since_registration* | Number of days since registration | Derived Feature |
| *is_cross_border* | Binary to check if the transaction is cross border or not | Countries.csv |
| *cross_border_to_high_risk* | Binary to check if the transaction is cross border to high risk country or not | |

## 2. RISK SCORING

Here, the task was to apply business rules in order to find suspicious transactions and anomalous clients. For this, the rules that have been stated in "*compliance_rules.csv"* have been considered. A total of 11 rules have been considered which are the business rules that have been applied in this assignment. All the rules are broken down into "CLIENT RULES" (Rules which are applied on anomalous clients) and "TRANSACTION RULES" (Rules which are applied to find suspicious transactions). For this assignment, all rules are set to ACTIVE however, the user can deactivate the rules according to their requirement in the *"compliance_rules.csv"*

| rule_name | rule_description | risk_weight |
|---|---|---|
| TRANSACTION RULE - High Transaction Amount | Flag transactions greater than 10,000 EUR as potentially high risk. | 0.8 |
| TRANSACTION RULE - High-Risk Country (Origin) | Flag transactions originating from high-risk countries. | 0.9 |
| TRANSACTION RULE - High-Risk Country (Destination) | Flag transactions sent to high-risk countries. | 0.9 |
| TRANSACTION RULE - Cross-Border to High-Risk Country | Transactions to high-risk countries from non-risky countries | 0.9 |
| TRANSACTION RULE - Very large transaction by withdrawal | Withdrawal amount of more than 10,000 EUR | 0.9 |
| CLIENT RULE - Multiple Failed Transactions | Flag clients with more than 3 failed transactions in the last 30 days. | 0.6 |
| CLIENT RULE - Frequent Large Transactions | Flag clients who have made more than 5 transactions >10,000 in the last 7 days. | 0.7 |
| CLIENT RULE - Document Verification Failed | Flag clients whose identification document failed verification. | 0.85 |
| CLIENT RULE - Frequent cross-border transactions | Clients who have made frequent transactions across the border in last 90 days | 0.6 |
| CLIENT RULE - Transactions to high-risk countries | Clients who have made transactions to high-risk countries | 0.9 |

The following rules have been applied in the script titled *"apply_business_rules.py"* which can be found in the folder titled *"rule_based_risk_scoring"*. The outputs consisting of the risk scores of all the clients, suspicious transactions and the triggered events can be found in the folder *"data\risk_scores"*.

Within "data/risk_scores", the CSV file titles *"client_risk_summary"* contains a list of all clients, their associated risk scores and the rules which have been triggered to assign the risk score. Similarly the CSV file titled *"transaction_risk_summary"* contains a list of all transactions, its risk score and a list of all triggered rules for each transaction which has lead to the assignment of the risk score. Risk score is calculated using the total weights corresponding to each rule which has been triggered.

## 3. ML BASED SCORING

For this part of the task, 2 different machine learning models were developed. The unsupervised learning method called "Isolation Forest" has been considered in order to design the models. Again, 2 different models have been proposed. One to find all the anomalous clients and another in order to find a list of suspicious clients. The codes for training each of the model as well as the features selected can be found in:

*a*. For Clients: *"/machine_learning_based_risk_scoring/isolation_forest_clients.py"*
*b*. For Transactions: *"/machine_learning_based_risk_scoring/isolation_forest_clients.py"*

Results that have been generated using purely the machine learning based method can be found in the folder titled "/data/kyc_ML_risk_scores". The CSV files also include the reason why the client/transaction is suspicious.

*a*. Results For Anomolous Clients from ML model:
*"/data/kyc_ML_risk_scores/client_ml_risk_scores.csv"*
*b*. Results for Suspicious Transactions from ML model:
*"/data/kyc_ML_risk_scores/transaction_ml_risk_scores_with_reason.csv"*

### 4. OUTPUT GENERATION

Importantly, the output has to be easy to be understandable by the end user (who might not be a developer). Thus, to generate output, we combine the scores coming from both ML based scoring method as well as risk score derived from purely applying the business rules. Again the script for combining and processing the output datasets can be found at:

*a*. For Clients: *"/output_generation/output_anomolous_clients.py"*
*b*. For Transactions: *"/output_generation/output_suspicious_transactions.py"*

Results which have been generated using these scripts stated above are the final output dataset which can be used by the user. The can be found in the folder titled "final_output_dataset" and can be found at:

*a*. Results For Anomolous Clients from ML and Business Rules with Reasons :
*"final output dataset/clients_combined_risk_scores.csv"*
*b*. Results for Suspicious Transactions from ML and Business Rules with Reasons:
*"final output dataset/transactions_combined_risk_scores.csv"*

*Note: ml_risk_score is the normalised value of the computed ml_anomaly_score. If ml_anomaly_flag = -1, the client/transaction is in medium/high risk and if ml_anomaly_flag=1, the client/transaction is low risk.*

**TASK 2.5: PIPELINE ORCHESTRATION**

**The entire pipeline can be run using the script titled *"main_pipeline.py".*** This script consists of the entire pipeline and entire code which is meant to generate the results and outputs. The same can also be done on Databricks jobs however, after the new Databricks update DBFS is deprecated when the **community edition** was changed to Free Edition few days, public DBFS is restricted. In Community or Free edition one only has access to serverless compute. In this serverless compute, access to legacy directory such as Filestore is not allowed and since I am using the free edition, I am not allowed to access this.

**TASK 2.6 Architecture**

The above figure depicts the architecture of how the solution would look on the cloud platform of Azure using various Azure tools as well as Databricks. The entire solution on cloud can be spread into a total of 6 main layers.

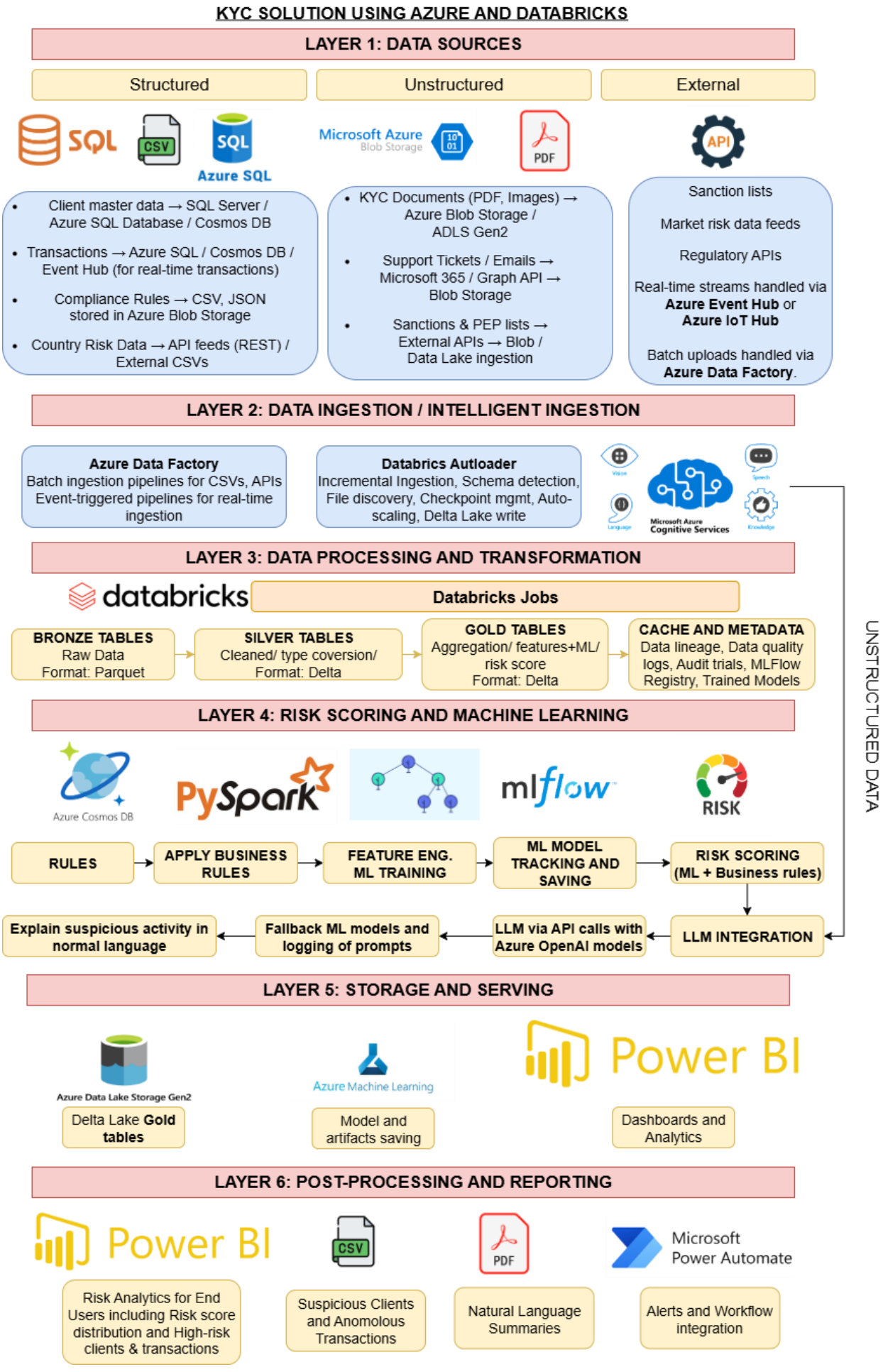
#### a. Data Sources

The user of the cloud model will be in touch with this layer. This layer consists of different type of documents, files etc. that the client submits during the KYC process. Additionally, this also consists of transactions and other important information which is useful in the further process. Based on the type of data, it can be stored in one of the options which has been given.

#### b. Data Ingestion/Intelligent Ingestion

The Data Ingestion Layer in this cloud-based KYC solution serves as the foundational entry point for all data entering the platform. It is designed to efficiently collect, validate, and store diverse data sources ranging from client profiles and transaction records to external compliance lists into a centralized and secure cloud environment. Using Azure Data Factory, data from various sources such as internal banking systems, APIs, and flat files (like CSVs or JSON) is ingested either in batch mode (scheduled data loads) or real-time streaming mode through services like Azure Event Hubs. The ingested data is then stored in Azure Data Lake Storage Gen2 (ADLS Gen2) as raw datasets, forming the Bronze layer of the data architecture. This raw storage follows a schema-on-read design, enabling flexibility in handling both structured and semi-structured data without enforcing strict schemas during ingestion. To ensure data reliability, automated validation and quality checks are performed through Azure Databricks notebooks, where missing values, data type mismatches, and schema inconsistencies are flagged and logged. Metadata about data lineage and source systems is captured in Azure Purview, supporting traceability and governance. The Data Ingestion Layer is inherently scalable and resilient, as Azure services automatically scale resources based on data volume and workload. New data sources can easily be added by configuring new pipelines or connectors without affecting existing workflows. Once data is ingested and validated, it becomes available for downstream transformation and enrichment in Databricks, feeding into the Silver and Gold layers of the architecture. This modular and cloud-native approach ensures that the ingestion

layer not only supports growing data demands but also maintains compliance, security, and operational efficiency across the entire KYC lifecycle.



### c. Data Processing and Transformation

In this KYC cloud architecture, Layer 3 which is Data Processing and Transformation represents the core data engineering and analytics engine where raw ingested data is refined, standardized, and enriched for downstream machine learning and reporting workflows. This layer is implemented primarily using **Azure Databricks**, leveraging **Apache Spark** for distributed data processing and transformation at scale. The process begins with the **Bronze layer**, where all raw data ingested from diverse structured and unstructured sources such as Azure SQL Database, Event Hubs, and Blob Storage is stored in Parquet format. This ensures cost-effective and schema-flexible data storage. Data is then transformed to **Silver layer**. In this stage, data undergoes cleaning, schema alignment, type conversion, and standardization. This standardization ensures that all data conforms to a consistent format suitable for analytics and modeling.

Once standardized, the data progresses to the **Gold layer**, which contains curated, aggregated, and feature-enriched datasets ready for machine learning and risk scoring. Here, PySpark and SQL-based transformations are used to derive key risk indicators, join client and transaction datasets, and compute metrics such as average transaction size, frequency, and geographic risk exposure. The transformation workflows are orchestrated using **Databricks Jobs** and integrated with **Azure Data Factory** for scheduling, monitoring, and dependency management. Metadata, data lineage, and data quality metrics are captured in **Azure Purview**, ensuring transparency and compliance with regulatory standards. The system also maintains a cache and metadata layer within Databricks that records audit trails, MLFlow model tracking, and data quality logs. Overall, this layer ensures that data flows seamlessly from raw ingestion to a fully prepared analytical and ML-ready state. It enables scalability, traceability, and reusability, allowing new features or risk models to be built rapidly without disrupting existing pipelines making it a robust foundation for advanced KYC analytics and AI-driven decision-making.

### d. Risk Scoring and Machine Learning

This layer is where the intelligence of the KYC system comes to life, it combines machine learning (ML) models and business rules to assess client and transaction risk dynamically. Built on the Azure Databricks and Azure Machine Learning ecosystem, it uses PySpark, MLflow, and Azure Cosmos DB to orchestrate, train, and operationalize ML-driven risk scoring at scale. Data from the Gold layer (processed and enriched data) flows into Databricks for feature engineering. Unstructured documents stored in Azure Blob Storage / ADLS Gen2 are processed using Azure Cognitive Services including Form Recognizer, Text Analytics, and Computer Vision to extract textual and semantic information. Extracted text (e.g., passport details, corporate ownership documents, or email correspondences) is cleaned and vectorized using Azure OpenAI Embeddings API before being passed to the LLM for contextual interpretation or summarization.

This hybrid ingestion enables the LLM to reason over both transactional patterns (structured) and contextual documents (unstructured), producing a comprehensive, human-readable risk narrative. Here, advanced ML algorithms (e.g., Isolation Forest, Gradient Boosting, or Neural Networks) are trained to identify anomalous transaction patterns or suspicious client behavior. The feature store in Databricks ensures consistency between training and inference by maintaining reusable, versioned features.

Business logic rules such as transactions exceeding certain thresholds, high-risk countries, or unusual activity frequency are applied on top of ML outputs. The rules are stored and maintained in Azure Cosmos DB, allowing dynamic updates without redeploying models. MLflow handles model lifecycle management including tracking experiments, logging parameters, and versioning models. Once trained, models are registered in the MLflow Registry and optionally deployed to Azure Machine Learning endpoints for scalable inference. The layer also integrates Azure OpenAI Service (LLM) through secure API calls, enabling explainability features such as translating complex anomaly detections into natural-language explanations. In this cloud-native setup, models are containerized and deployed via Azure Kubernetes Service (AKS) or Azure Machine Learning Managed Endpoints for production-grade scalability, security, and low-latency predictions. The results risk scores and categories are then stored back into Delta Lake for downstream analytics. In this architecture, the LLM is accessed securely via the Azure OpenAI Service API, rather than hosted internally.

### e. Storage and Serving

This layer acts as the centralized and secure data serving and model storage zone of the KYC solution. It ensures that both data and ML artifacts are persisted, versioned, and made accessible to downstream systems such as dashboards, APIs, or alerting workflows. The primary data store is Azure Data Lake Storage Gen2, organized into Delta Lake tables (Bronze, Silver, and Gold). These Gold tables contain final aggregated and scored datasets including ML-based and business-rule-based risk scores for

each client and transaction. Delta Lake's support for ACID transactions and schema evolution allows for real-time updates, incremental loading, and reproducibility.

All ML models, metadata, and artifacts are securely stored in Azure Machine Learning (AML)'s model registry, integrated with MLflow. This allows teams to deploy, monitor, and retrain models without manual intervention. For real-time use cases, these models can be hosted as REST APIs via Azure Machine Learning Managed Endpoints or AKS, ensuring low-latency inference for live transaction scoring. The layer is also responsible for managing access control and governance using Azure Active Directory (AAD) and Azure Role-Based Access Control (RBAC). It guarantees that sensitive KYC data including client identities and transaction details are encrypted in transit (TLS) and at rest (using Azure-managed encryption keys). Ultimately, this layer bridges analytical processing with real-world usability, ensuring secure, high-performance data serving and model access across multiple Azure services.

### f. Post Processing and Reporting

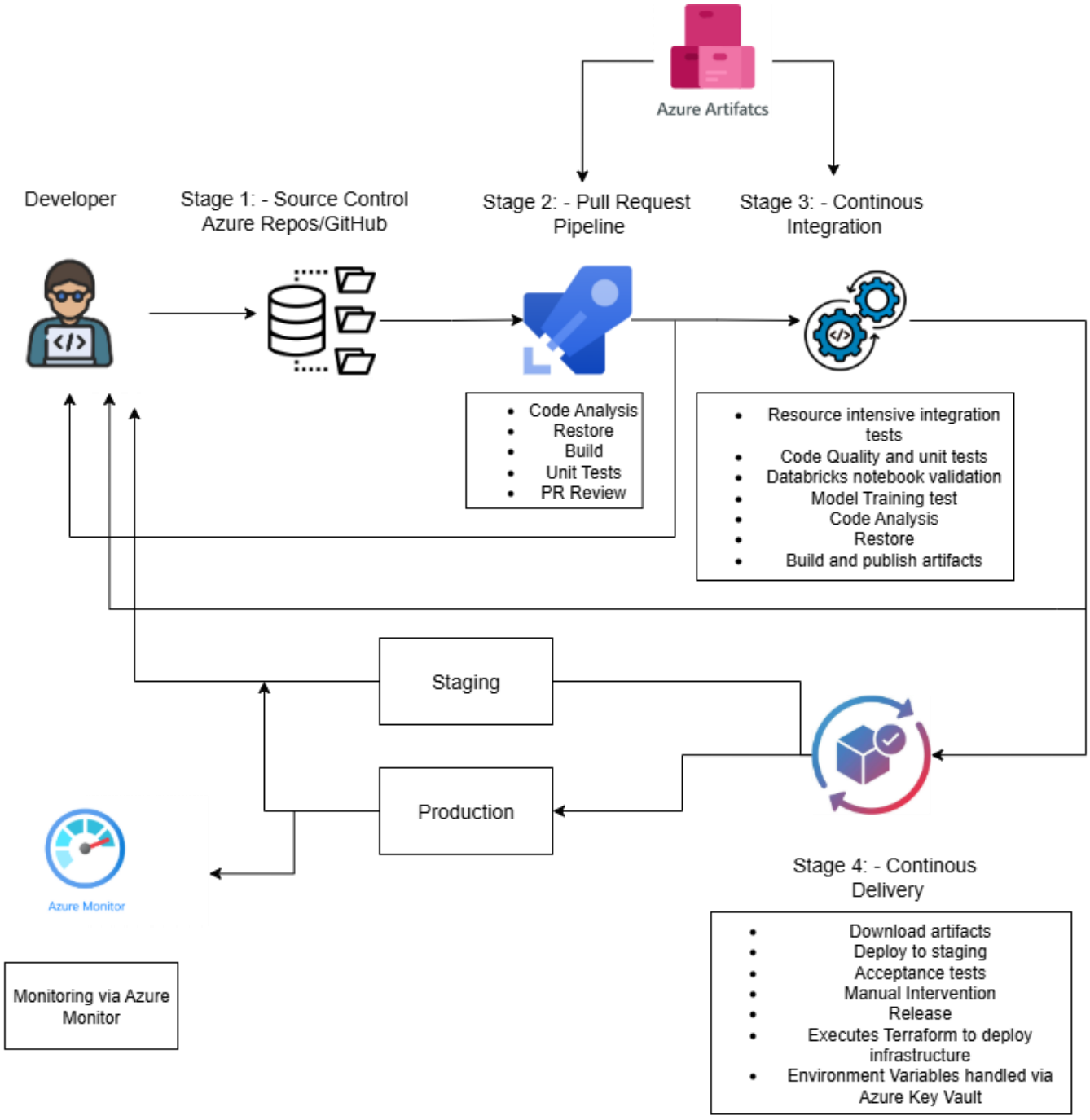
The Post-Processing and Reporting layer transforms analytical results and risk scores into actionable insights for compliance officers, analysts, and business stakeholders. It leverages Power BI, Microsoft Power Automate, and Azure Synapse Analytics to deliver intelligent, automated, and interactive reporting in the cloud.

The scored datasets (stored in Delta Lake Gold tables) are connected directly to Power BI through Azure Synapse Serverless SQL or Databricks SQL endpoints, enabling live dashboards. These dashboards visualize key KYC metrics such as high-risk client distributions, anomaly trends, and transaction risk scores with drill-down capabilities by region, client type, or transaction category. To enhance compliance workflows, Microsoft Power Automate integrates with this layer to trigger alerts, notifications, or case-creation workflows. For instance, if a client's risk score exceeds a predefined threshold, a workflow automatically generates an alert or a compliance review task in tools like Microsoft Teams or ServiceNow.

The outputs generated by the LLM are routed to various downstream consumers for compliance and decision-making:

- Power BI Dashboards: Dashboards updated live which showcase the updating of the risk scores.
- Regulatory and Audit Reports (PDF/CSV): LLM outputs are formatted into PDF or CSV reports using Databricks or Azure Logic Apps, automatically stored in Blob Storage for archiving and audit trails.
- Workflow Systems (Microsoft Power Automate / Teams): Power Automate triggers alerts when the LLM identifies severe anomalies or risk patterns. The summarized insight is posted directly into a compliance channel or case management system.
- APIs for External Consumption: Risk summaries and narrative explanations can be served via Azure API Management (APIM), enabling external regulators or internal risk systems to query and consume LLM-enriched insights programmatically.

**TASK 2.7 CI/CD Pipeline (Optional/Bonus)**




**Task 2.8 Agent Based Extension (Optional/Bonus)**

Core Concept:

A brief prototype as asked in the assignment has been implemented where multiple specialized agents work together to evaluate clients and make decisions about their risk level. Rather than having one monolithic decision engine, different agents focus on different aspects of the evaluation, then their findings are combined into a final verdict. The Five Agents are as follows:

- Data Validator Agent - Checks if required fields are present and not null (client_id, name, country, kyc_status, email). It flags data quality issues and reduces confidence for each missing field.
- Risk Analyzer Agent - Combines ML risk scores with business rule scores into a combined score. Makes decisions based on thresholds: >70 = ESCALATE, >50 = REVIEW, otherwise APPROVE.
- Compliance Checker Agent - Enforces regulatory rules: rejects clients from restricted countries (Nigeria, Russia) or those with unverified KYC status. Simple hard rules.

- Investigator Agent - Deep dives into specific anomalies only when needed: cross-border transactions to high-risk countries, multiple failed transactions, or unusual patterns like 8+ large transactions in 7 days. Only runs if the Risk Analyzer or Compliance Checker flags concerns.
- Decision Maker Agent - Synthesizes all agent decisions into a final verdict. It counts votes: if any agent says REJECT, it rejects. If 2+ agents say ESCALATE, it escalates. Otherwise, it reviews or approves based on remaining votes.

The code for this multi-agent problem can be found in the file titled *"multi_agent.py"* and the results can be seen in the terminal where client 1 is accepted and client 2 gets rejected. The results can only be obtained after evaluation of "main_pipeline.py" file.