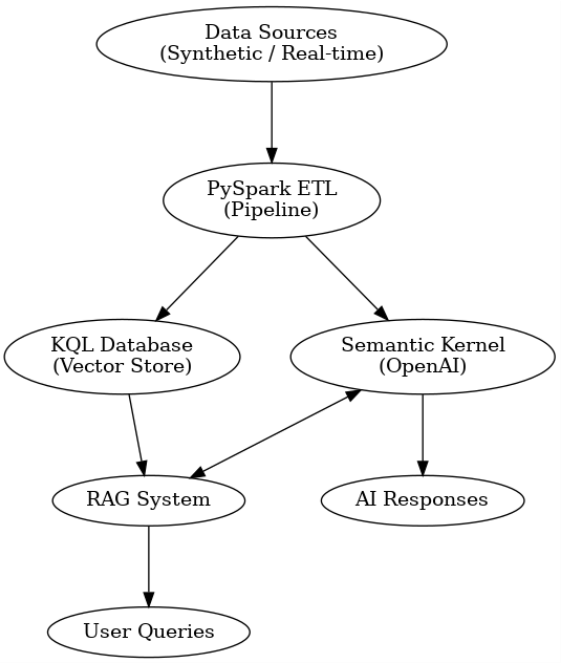
LTA Asset Management Analytics- Technical Documentation

Table of Contents

System Architecture Overview

High-Level Architecture



Components Description

1. **Data Ingestion Layer**: PySpark-based ETL pipeline for asset data
2. **Vector Storage**: KQL Database for storing embeddings and metadata
3. **AI Orchestration**: Semantic Kernel for LLM interactions
4. **RAG Engine**: Retrieval-Augmented Generation system for asset management

---

Data Pipeline Design

Step 1: Synthetic Data Generation

```
def create_lta_asset_synthetic(num_records: int = 1000)-> DataFrame:
  """
  Generates synthetic asset management data with:
  - Asset identification (ID, type, manufacturer)
  - Condition monitoring metrics
  - Maintenance history
  - Risk assessment scores
  - Performance indicators
  """
```

Key Data Entities:

- **Asset Metadata**: ID, type, line, station, manufacturer
- **Temporal Data**: Installation date, age, maintenance intervals
- **Condition Metrics**: Temperature, vibration, power consumption
- **Risk Scores**: Failure probability, condition score, risk assessment
- **Maintenance Data**: Cost, duration, priority, type

Step 2: Feature Engineering

```
# Asset Management Specific Features
.withColumn("maintenance_efficiency",
        round(col("availability_percentage") / (col("maintenance_cost") / 1000 + 1), 2))
.withColumn("asset_utilization",
        round((100- col("downtime_minutes") / 1440 * 100), 2))
.withColumn("maintenance_urgency_index",
        round(col("risk_score") * col("failure_probability") * 100, 2))
```

Engineered Features:

1. **Maintenance Efficiency**: Availability per maintenance cost
2. **Asset Utilization**: Operational time percentage
3. **Risk Alerts**: Overheating, vibration thresholds
4. **Cost Metrics**: Cost per availability year
5. **Predictive Indicators**: Failure probability forecasts

Step 3: Context Generation for LLM

```
lta_enriched_df = lta_enriched_df.withColumn(
   "asset_context",
   concat_ws(" | ",
      col("asset_id"),
      col("asset_type"),
      col("line"),
      col("station"),
      concat_ws(", ",
         concat_ws("=", lit("Age"), col("asset_age_years")),
         concat_ws("=", lit("Condition"), col("condition_state")),
         concat_ws("=", lit("Risk"), col("risk_score")),
         # ... additional metrics
      )
   )
)
```

Semantic Kernel Integration

Initialization

```
class LTA_ASSET_RAG_System:
   def __init__(self, spark_df: DataFrame, openai_api_key: str):
      self.kernel = sk.Kernel()

      # Chat Completion Service
      self.chat_service = OpenAIChatCompletion(
         service_id="chat_completion",
         ai_model_id=OPENAI_CHAT_MODEL,
         api_key=openai_api_key
      )
      self.kernel.add_service(self.chat_service)

      # Embedding Service
      self.embedding_service = OpenAITextEmbedding(
         service_id="text_embedding",
         ai_model_id=OPENAI_EMBEDDING_MODEL,
         api_key=openai_api_key
      )
      self.kernel.add_service(self.embedding_service)
```

Configuration

```
# Model Configuration
OPENAI_CHAT_MODEL = "gpt-4"
OPENAI_EMBEDDING_MODEL = "text-embedding-ada-002"

# Execution Settings
settings = self.chat_service.get_prompt_execution_settings_class()(
    temperature=0.3,      # Lower for consistent technical responses
    max_tokens=1000,      # Adequate for detailed analysis
    top_p=0.9
)
```

Services Utilization

1. **Embedding Generation**:
    - Converts asset context to vector representations
    - Uses text-embedding-ada-002 for high-quality embeddings
    - Batch processing for efficiency
2. **Chat Completion**:
    - GPT-4 for complex asset management reasoning
    - Structured prompts for consistent output format
    - Error handling for API limitations

KQL Database as Vector Store

Schema Design

```
// AssetTelemetry Table Schema
.create-merge table AssetTelemetry (
    Timestamp: datetime,
    AssetId: string,
    AssetType: string,
    Line: string,
    Station: string,
    Manufacturer: string,
    AssetAgeYears: int,
    ConditionScore: real,
    ConditionState: string,
    RiskScore: real,
    FailureProbability: real,
    RemainingUsefulLife: real,
    MaintenancePriority: string,
    MaintenanceCost: real,
    AvailabilityPercentage: real,
    AssetContext: string,
    AssetMetricsVector: dynamic,  // Vector embeddings
    MaintenanceEfficiency: real,
    AssetUtilization: real
)
```

Vector Storage Strategy

```
# Embedding Generation UDF
@pandas_udf(ArrayType(FloatType()))
def embed_asset_batch(texts: pd.Series)-> pd.Series:
    client = OpenAI(api_key=OPENAI_API_KEY)
    response = client.embeddings.create(
        model="text-embedding-3-small",
```

```
        input=texts.tolist()
    )
    vectors = [d.embedding for d in response.data]
    return pd.Series(vectors)

# Store in KQL
lta_enriched_df = lta_enriched_df.withColumn(
    "asset_metrics_vector", embed_asset_batch(col("asset_context"))
)
```

KQL Ingestion Configuration

```
kustoOptions = {
    "kustoCluster": KQL_CLUSTER,
    "kustoDatabase": KQL_DB,
    "kustoTable": KQL_TABLE,
    "kustoIngestionType": "queued"  # Alternatives: "streaming"
}

lta_enriched_df.write \
    .format("com.microsoft.kusto.spark.synapse.datasource") \
    .options(**kustoOptions) \
    .option("accessToken", access_token) \
    .mode("append") \
    .save()
```

RAG Implementation

Memory Initialization

```
def _initialize_memory(self):
    """Load high-risk assets into in-memory store for quick RAG access"""
    high_risk_assets = self.df.filter(
        (col("risk_score") > 0.7) |
        (col("condition_state").isin(["Poor", "Critical"])) |
        (col("maintenance_priority") == "High") |
        (col("failure_probability") > 0.6)
    ).orderBy(col("risk_score").desc()).limit(200).collect()
```

Semantic Search Algorithm

```
def vector_search(self, query: str, top_k: int = 5)-> List[Dict]:
    """
    Asset-specific semantic search using risk and condition heuristics
    """
    q = query.lower()

    # Domain-specific search conditions
    if "risk" in q or "failure" in q or "critical" in q:
        condition = col("risk_score") > 0.7
    elif "maintenance" in q or "repair" in q:
        condition = col("maintenance_priority").isin(["High", "Medium"])
    elif "condition" in q or "health" in q:
        condition = col("condition_state").isin(["Poor", "Critical"])
    # ... additional conditions

    return self.df.filter(condition).orderBy(col("risk_score").desc()).limit(top_k).collect()
```

Prompt Engineering

```
def augment_prompt(self, query: str, context: List[Dict])-> str:
    """
    Constructs domain-specific prompt for asset management analysis
    """
    enhanced = f"""You are an expert asset management AI assistant for Land Transport Authority (LTA).

    HIGH-RISK / CRITICAL ASSET RECORDS:
    {context_str}

    USER QUERY: {query}

    Please provide comprehensive asset management analysis:

    1) RISK ASSESSMENT: Evaluate overall risk exposure and critical assets
    2) MAINTENANCE STRATEGY: Recommend maintenance approaches
    3) COST OPTIMIZATION: Suggest cost-effective maintenance scheduling
    4) LIFECYCLE MANAGEMENT: Provide replacement/renewal recommendations
    5) DATA-DRIVEN INSIGHTS: Highlight patterns and predictive opportunities

    Focus on operational efficiency, safety, and cost-effectiveness."""
```

Async RAG Workflow

```
async def rag_workflow_async(self, query: str, top_k: int = 5)-> Dict[str, Any]:
    """
    Complete RAG pipeline execution
    """
    print(f" 🔍 Processing asset management query: '{query}'")

    # Step 1: Context Retrieval
    relevant_context = self.vector_search(query, top_k=top_k)

    # Step 2: Prompt Augmentation
    enhanced_prompt = self.augment_prompt(query, relevant_context)

    # Step 3: AI Response Generation
    response = await self.generate_response_async(enhanced_prompt)

    return {
        "query": query,
        "context": relevant_context,
        "enhanced_prompt": enhanced_prompt,
        "response": response
    }
```

Deployment Guide

Prerequisites

```
# Required Packages
pip install semantic-kernel
pip install openai
pip install pyspark
pip install pandas
pip install numpy
```

```
# Azure/AWS Dependencies
pip install azure-kusto-data
pip install azure-kusto-ingest
```

Environment Configuration

```
# Environment Variables
import os
os.environ["OPENAI_API_KEY"] = "your-api-key-here"
os.environ["KUSTO_CLUSTER"] = "https://your-cluster.kusto.fabric.microsoft.com"
os.environ["KUSTO_DATABASE"] = "LTA_Asset_Management_DB"

# Spark Configuration
spark = SparkSession.builder \
    .appName("LTA_Asset_Management_RAG") \
    .config("spark.sql.adaptive.enabled", "true") \
    .config("spark.sql.adaptive.coalescePartitions.enabled", "true") \
    .getOrCreate()
```

Deployment Steps

1. **Data Pipeline Deployment**:

```
# Schedule synthetic data generation
spark-submit--master yarn data_pipeline.py
```

2. **KQL Database Setup**:

```
.create database LTA_Asset_Management_DB
.create table AssetTelemetry
.alter table AssetTelemetry policy streamingingestion enable
```

3. **RAG Service Deployment**:

```
# Initialize and test the system
rag_system = LTA_ASSET_RAG_System(lta_asset_df, OPENAI_API_KEY)
result = await rag_system.rag_workflow_async("high risk assets")
```

Monitoring and Logging

```
# Add monitoring decorators
def log_rag_workflow(func):
    def wrapper(*args, **kwargs):
        start_time = datetime.now()
        result = func(*args, **kwargs)
        duration = datetime.now()- start_time
        print(f"RAG workflow completed in {duration.total_seconds():.2f}s")
        return result
    return wrapper
```

API Reference

LTA_ASSET_RAG_System Class

Methods

__init__(spark_df: DataFrame, openai_api_key: str)

- Initializes the RAG system with data and API credentials

vector_search(query: str, top_k: int = 5)-> List[Dict]

- Performs semantic search on asset data
- Returns relevant asset records based on query

augment_prompt(query: str, context: List[Dict])-> str

- Enhances user query with retrieved context
- Returns formatted prompt for LLM

generate_response_async(enhanced_prompt: str)-> str

- Asynchronously generates AI response
- Returns structured asset management analysis

rag_workflow_async(query: str, top_k: int = 5)-> Dict[str, Any]

- Complete RAG pipeline execution
- Returns query, context, prompt, and response

Usage Example

```
# Initialize system
rag = LTA_ASSET_RAG_System(asset_data, "your-openai-key")

# Execute RAG query
result = await rag.rag_workflow_async(
   "Identify assets with high failure probability on East-West line",
   top_k=5
)

print(result["response"])
```

Data Models

Asset Record Schema

```
{
   "asset_id": str,
   "asset_type": str,
   "line": str,
   "station": str,
   "manufacturer": str,
   "asset_age_years": int,
   "condition_score": float,
   "condition_state": str,
   "risk_score": float,
   "failure_probability": float,
   "remaining_useful_life": float,
   "maintenance_priority": str,
   "maintenance_cost": float,
   "availability_percentage": float
}
```

RAG Response Schema

```
{
   "query": str,
   "context": List[AssetRecord],
   "enhanced_prompt": str,
   "response": str
}
```

Performance Considerations

Optimization Strategies

1. Vector Search Optimization:
    - Implement approximate nearest neighbor (ANN) search in KQL
    - Use cosine similarity for embedding comparisons
    - Cache frequently accessed vectors
2. Memory Management:
    - Limit in-memory store to high-priority assets
    - Implement LRU cache for context retrieval

- Use streaming ingestion for real-time updates

3. API Rate Limiting:
   - Implement exponential backoff for OpenAI API calls
   - Batch embedding generation requests
   - Cache LLM responses for similar queries

Scaling Recommendations

1. Horizontal Scaling:
   - Deploy multiple RAG service instances
   - Use load balancer for query distribution
   - Implement database connection pooling
2. Data Partitioning:
   - Partition KQL tables by asset type and line
   - Use time-based partitioning for temporal queries
   - Implement geographic partitioning by station

This documentation provides a comprehensive guide to implementing the **LTA Asset Management Analytics** system using **Semantic Kernel, KQL Database, and RAG architecture**. The system is designed to be scalable, maintainable, and directly aligned with the asset management use cases.

Outputs

```
================================================================================
DATA ENRICHMENT AND TRANSFORMATION - LTA ASSET MANAGEMENT
================================================================================
✓ Asset management features added (efficiency, utilization, risk alerts, cost metrics).
✓ Added 'asset_context' column for LLM prompt enrichment.
✓ Added 'asset_metrics_vector' column.

================================================================================
LOAD DATA TO KQL DATABASE - LTA ASSET MANAGEMENT
================================================================================
✓ Asset data prepared for KQL ingestion
  Records to ingest: 2,000
✓ Data saved to Delta table: LTA_asset_management_staging
Total asset records prepared for ingestion: 2000
+----------+------------------+------------------+---------------+----------+--------------------+
|asset_id  |asset_type        |line              |condition_state|risk_score|maintenance_priority|
+----------+------------------+------------------+---------------+----------+--------------------+
|ASSET-0025|Ventilation_System|Thomson-East Coast|Fair           |0.783     |High                |
|ASSET-0130|Signaling_System  |North-East        |Fair           |0.774     |High                |
|ASSET-0093|CCTV_Camera       |Circle            |Fair           |0.372     |Medium              |
|ASSET-0116|Escalator         |Thomson-East Coast|Excellent      |0.629     |Medium              |
|ASSET-0128|Escalator         |Circle            |Critical       |0.28      |High                |
+----------+------------------+------------------+---------------+----------+--------------------+
  → Initializing asset management memory...
  ✓ Loaded 200 high-risk asset records into memory
✓ LTA Asset Management RAG System initialized with OpenAI.
  Chat Model: gpt-3.5-turbo
  Embedding Model: text-embedding-ada-002

============================================================
QUERY: high risk assets requiring immediate maintenance
============================================================
  → Performing asset semantic search for: 'high risk assets requiring immediate maintenance'
  ✓ Found 3 relevant asset records

Top 3 relevant assets:
1. ASSET-0169 | Power_Supply | Condition: Poor | Risk: 0.907 | Priority: High
2. ASSET-0092 | Signaling_System | Condition: Good | Risk: 0.905 | Priority: High
3. ASSET-0080 | Ticketing_Machine | Condition: Good | Risk: 0.902 | Priority: High

ASSET MANAGEMENT ANALYSIS:
Based on the high-risk assets requiring immediate maintenance, here is a comprehensive asset management analysis:
```

1) RISK ASSESSMENT:
   - Overall Risk Exposure: The three high-risk assets identified have risk scores ranging from 0.902 to 0.907, indicating a significant level of risk.
   - Critical Assets: ASSET-0169 (Power Supply) and ASSET-0092 (Signaling System) are critical assets with high failure probabilities and relatively low remaining useful life (RUL).

2) MAINTENANCE STRATEGY:
   - Predictive Maintenance: Implement predictive maintenance for ASSET-0169 and ASSET-0092 to monitor their condition in real-time and anticipate potential failures.
   - Preventive Maintenance: Conduct regular preventive maintenance for ASSET-0080 to ensure its continued good condition and extend its RUL.

3) COST OPTIMIZATION:
   - Prioritize Maintenance: Allocate resources to address ASSET-0169 and ASSET-0092 first due to their higher risk levels and criticality.
   - Cost-Effective Maintenance: Optimize maintenance schedules to minimize downtime and maximize asset availability while controlling costs.

4) LIFECYCLE MANAGEMENT:
   - Replacement Recommendations: Consider replacing ASSET-0169 and ASSET-0092 within the next few years as their RULs are relatively short, and their conditions are deteriorating.
   - Renewal Strategies: Evaluate the feasibility of upgrading or renewing the assets to improve their performance and reliability.

5) DATA-DRIVEN INSIGHTS:
   - Patterns and Predictive Maintenance: Analyze historical data to identify patterns of asset failures and develop predictive maintenance models for early detection of issues.
   - Continuous Monitoring: Implement continuous monitoring systems for critical assets to capture real-time data and enable proactive maintenance actions.

By focusing on operational efficiency, safety, and cost-effectiveness, the recommended maintenance strategies and lifecycle management approaches aim to ensure the reliability and longevity of the high-risk assets in the LTA's transportation network.
==

==
QUERY: assets with poor condition scores on North-South line
==
  → Performing asset semantic search for: 'assets with poor condition scores on North-South line'
  ✓ Found 3 relevant asset records

Top 3 relevant assets:
1. ASSET-0169 | Power_Supply | Condition: Poor | Risk: 0.907 | Priority: High
2. ASSET-0062 | Ventilation_System | Condition: Poor | Risk: 0.901 | Priority: High
3. ASSET-0163 | Power_Supply | Condition: Poor | Risk: 0.901 | Priority: High

ASSET MANAGEMENT ANALYSIS:
Based on the user query for assets with poor condition scores on the North-South line, the following assets fall under this category:

• ASSET-0062 (Ventilation_System) | Line: North-South Station: STN-068 | Age: 10y, Condition: Poor (Score: 1.24), Risk: 0.901, Failure Prob: 0.918, RUL: 1.9y, Priority: High, Cost: $43,805.84, Availability: 90.5%

To provide a comprehensive asset management analysis, we will address the user's query and focus on the following key areas:

1) RISK ASSESSMENT:
The overall risk exposure on the North-South line is significant due to the presence of assets with poor condition scores. The asset ASSET-0062 (Ventilation_System) stands out as a critical asset with a high risk score of 0.901 and a high probability of failure at 0.918. It is essential to address this asset promptly to mitigate potential disruptions and safety hazards.

2) MAINTENANCE STRATEGY:
Given the poor condition and high risk associated with ASSET-0062, a proactive maintenance approach is recommended. Implementing a predictive maintenance strategy utilizing IoT sensors and data analytics can help monitor the asset's health in real-time and predict potential failures before they occur. This approach can increase asset reliability, reduce downtime, and extend the asset's remaining useful life.

3) COST OPTIMIZATION:
To optimize maintenance costs, it is crucial to prioritize maintenance activities based on asset criticality and risk. Allocating resources efficiently to address high-risk assets like ASSET-0062 can prevent costly breakdowns and minimize maintenance expenses in the long run. Additionally, considering the cost of replacement versus repair for assets nearing the end of their useful life can help optimize spending.

4) LIFECYCLE MANAGEMENT:
For ASSET-0062, with a remaining useful life of 1.9 years, it is essential to start planning for its replacement or renewal. Conducting a lifecycle cost analysis to compare the costs of continued maintenance versus replacement can inform decision-making and ensure optimal asset performance and reliability.

5) DATA-DRIVEN INSIGHTS:
By analyzing historical maintenance data and asset performance metrics, patterns and trends can be identified to predict future maintenance needs and optimize maintenance schedules. Leveraging predictive maintenance algorithms and machine learning models can help identify early warning signs of potential failures and enable proactive maintenance interventions.

In conclusion, addressing assets with poor condition scores on the North-South line, such as ASSET-0062, requires a proactive and data-driven approach to ensure operational efficiency, safety, and cost-effectiveness. By prioritizing maintenance activities, implementing predictive maintenance strategies, and planning for asset replacement, LTA can enhance asset reliability and minimize risks associated with critical assets.
==

==
QUERY: cost optimization for maintenance scheduling
==
  → Performing asset semantic search for: 'cost optimization for maintenance scheduling'
  ✓ Found 3 relevant asset records

Top 3 relevant assets:
1. ASSET-0169 | Power_Supply | Condition: Poor | Risk: 0.907 | Priority: High
2. ASSET-0092 | Signaling_System | Condition: Good | Risk: 0.905 | Priority: High
3. ASSET-0080 | Ticketing_Machine | Condition: Good | Risk: 0.902 | Priority: High

ASSET MANAGEMENT ANALYSIS:
Based on the high-risk/critical asset records provided, here is a comprehensive analysis for cost optimization for maintenance scheduling:

1) RISK ASSESSMENT:
- Overall risk exposure: The assets ASSET-0169 (Power_Supply) and ASSET-0092 (Signaling_System) have higher risk scores and failure probabilities, indicating a greater likelihood of failure. ASSET-0080 (Ticketing_Machine) has a lower risk score but still requires attention due to its high priority and cost.

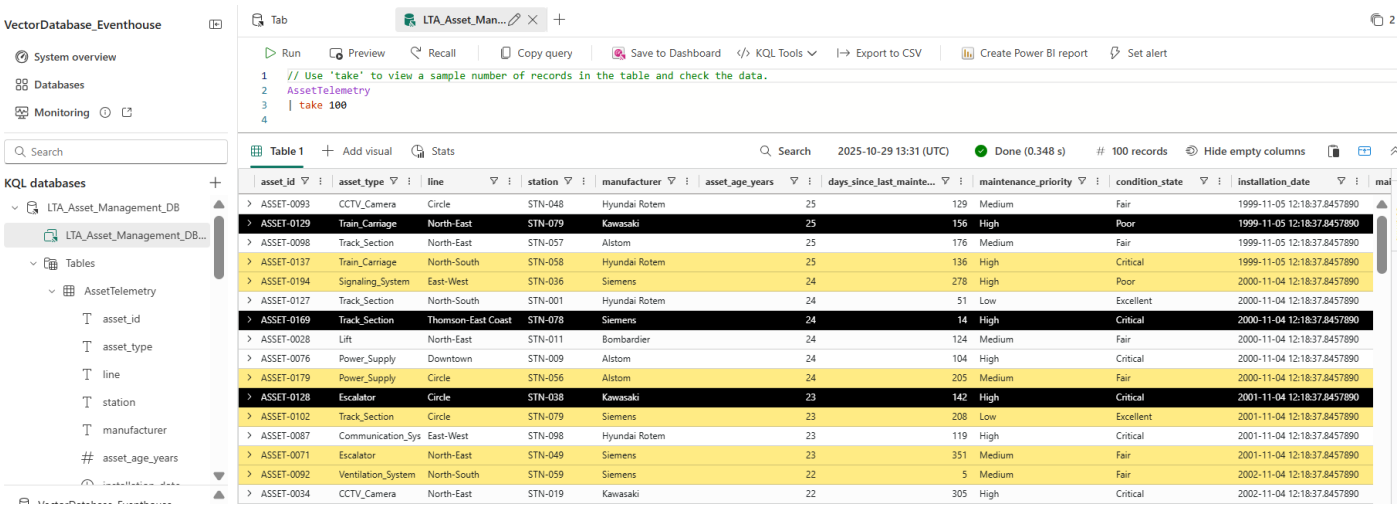
2) MAINTENANCE STRATEGY:
- Predictive maintenance: Implement predictive maintenance for ASSET-0169 and ASSET-0092 to proactively monitor and address potential issues before they lead to failures.
- Preventive maintenance: Continue with preventive maintenance for ASSET-0080 to ensure its good condition is maintained and to extend its remaining useful life.

3) COST OPTIMIZATION:
- Schedule maintenance activities for ASSET-0169 and ASSET-0092 based on their remaining useful life (RUL) and risk levels to optimize costs and minimize downtime.
- Consider bundling maintenance tasks for multiple assets in the same area to reduce travel time and maximize resource efficiency.

4) LIFECYCLE MANAGEMENT:
- Plan for the replacement/renewal of ASSET-0169 and ASSET-0092 before their RUL expires to avoid costly failures and disruptions.
- Evaluate the cost-effectiveness of renewing ASSET-0080 versus replacing it with a newer, more efficient ticketing machine.

5) DATA-DRIVEN INSIGHTS:
- Analyze historical maintenance data for patterns and trends to identify opportunities for predictive maintenance interventions.
- Utilize real-time monitoring and sensor data to detect early warning signs of potential failures and prioritize maintenance actions accordingly.

By focusing on these recommendations and leveraging data-driven insights, LTA can enhance operational efficiency, ensure safety, and achieve cost-effectiveness in asset management and maintenance scheduling.
==



asset_id	asset_type	line	station	manufacturer	asset_age_years	days_since_last_mainte...	maintenance_priority	condition_state	installation_date
ASSET-0093	CCTV_Camera	Circle	STN-048	Hyundai Rotem	25	129	Medium	Fair	1999-11-05 12:18:37.8457890
ASSET-0129	Train_Carriage	North-East	STN-079	Kawasaki	25	156	High	Poor	1999-11-05 12:18:37.8457890
ASSET-0098	Track_Section	North-East	STN-057	Alstom	25	176	Medium	Fair	1999-11-05 12:18:37.8457890
ASSET-0137	Train_Carriage	North-South	STN-058	Hyundai Rotem	25	136	High	Critical	1999-11-05 12:18:37.8457890
ASSET-0194	Signaling_System	East-West	STN-036	Siemens	24	278	High	Poor	2000-11-04 12:18:37.8457890
ASSET-0127	Track_Section	North-South	STN-001	Hyundai Rotem	24	51	Low	Excellent	2000-11-04 12:18:37.8457890
ASSET-0169	Track_Section	Thomson-East Coast	STN-078	Siemens	24	14	High	Critical	2000-11-04 12:18:37.8457890
ASSET-0028	Lift	North-East	STN-011	Bombardier	24	124	Medium	Fair	2000-11-04 12:18:37.8457890
ASSET-0076	Power_Supply	Downtown	STN-009	Alstom	24	104	High	Critical	2000-11-04 12:18:37.8457890
ASSET-0179	Power_Supply	Circle	STN-056	Alstom	24	205	Medium	Fair	2000-11-04 12:18:37.8457890
ASSET-0128	Escalator	Circle	STN-038	Kawasaki	23	142	High	Critical	2001-11-04 12:18:37.8457890
ASSET-0102	Track_Section	Circle	STN-079	Siemens	23	208	Low	Excellent	2001-11-04 12:18:37.8457890
ASSET-0087	Communication_Sys	East-West	STN-098	Hyundai Rotem	23	119	High	Critical	2001-11-04 12:18:37.8457890
ASSET-0071	Escalator	North-East	STN-049	Siemens	23	351	Medium	Fair	2001-11-04 12:18:37.8457890
ASSET-0092	Ventilation_System	North-South	STN-059	Siemens	22	5	Medium	Fair	2002-11-04 12:18:37.8457890
ASSET-0034	CCTV_Camera	North-East	STN-019	Kawasaki	22	305	High	Critical	2002-11-04 12:18:37.8457890