Kuok Group Singapore AI & Data Engineering Platform: Comprehensive Documentation

Executive Overview

This documentation details the implementation of an **AI-powered Commodities Trading Intelligence Platform** for Kuok Group Singapore, leveraging Microsoft Fabric, Semantic Kernel, RAG (Retrieval-Augmented Generation), Eventhouse as a Vector Database and KQL (Kusto Query Language) to transform raw commodities data into strategic business insights.

---

1. Architecture Overview

1.1 System Architecture Diagram

```
Natural Language Query
       ↓
[Semantic Kernel + OpenAI]
       ↓
   [RAG System]
   /       \
[Vector Search]  [Text-to-KQL]
   |             |
[Commodities Data] [KQL Database]
       ↓
[Business Intelligence]
       ↓
Strategic Recommendations
```

1.2 Technology Stack

Component	Technology	Purpose
Data Processing	PySpark, Microsoft Fabric	Large-scale data transformation
AI/ML	Semantic Kernel, OpenAI GPT-4	Natural language processing
Vector Database	KQL Database	Semantic search and storage
Query Engine	Kusto Query Language	Analytical queries
Orchestration	Python, Async/Await	Pipeline management

---

2. Core Concepts Explained

2.1 RAG (Retrieval-Augmented Generation)

What is RAG?

RAG enhances LLM responses by retrieving relevant information from a knowledge base before generating answers.

Kuok Group Implementation:

# Traditional LLM vs RAG

Traditional: User Question → LLM → Generic Answer

RAG: User Question → Vector Search → Relevant Data + LLM → Contextual Answer

Business Value:

- **Accuracy**: Answers based on actual trading data
- **Relevance**: Context-aware responses specific to commodities
- **Transparency**: Can trace insights back to source data

2.2 Semantic Kernel

What is Semantic Kernel?

Microsoft's orchestration framework for AI applications that connects LLMs, memory, and business logic.

Key Components:

- **Kernel**: Central orchestrator
- **Plugins**: Reusable AI functions
- **Memory**: Vector-based storage
- **Planners**: AI task decomposition

Kuok Group Usage:

```
# Semantic Kernel setup for commodities trading
kernel = sk.Kernel()
kernel.add_chat_service("commodities_ai", OpenAIChatCompletion(...))
```

2.3 KQL (Kusto Query Language)

What is KQL?

A query language optimized for big data analytics in Azure Data Explorer.

Why KQL for Kuok Group?

- **Performance**: Handles billions of commodities records
- **Real-time**: Stream processing capabilities
- **Integration**: Native to Microsoft Fabric
- **Security**: Enterprise-grade governance

Example KQL Query:

```
kql
CommoditiesTrading
| where Commodity == "Crude Palm Oil (CPO)"
| where Country in ("Indonesia", "Malaysia")
| summarize
   AvgPrice = avg(PurchasePrice),
   TotalVolume = sum(ProductionVolume)
by Country, Supplier
| order by AvgPrice desc
```

2.4 Vector Databases & Embeddings

Concept:

Convert text/data into numerical vectors that capture semantic meaning.

Kuok Group Application:

```
# Converting trading context to vectors
trading_context = "Palm Oil from Indonesia- Quality 9.2- Price $780"
embedding = embed_text_batch(trading_context)  # → [0.23, -0.45, 0.89, ...]
```

Business Benefit:

- Find similar suppliers based on multiple criteria
- Semantic search beyond keyword matching
- Cluster analysis of trading patterns

---

3. Data Model & Schema

3.1 Commodities Trading Schema

```
commodities_schema = {
   "timestamp": "datetime",          # Transaction timestamp
   "commodity": "string",            # Crude Palm Oil, Palm Kernel Oil, etc.
   "country": "string",              # Indonesia, Malaysia, Thailand
   "region": "string",               # Sumatra, Sabah, Southern
   "supplier": "string",             # PT Sawit Makmur, Borneo Harvest, etc.
   "production_volume_metric_tons": "float",
   "purchase_price_usd_per_ton": "float",
   "market_price_usd_per_ton": "float",
   "quality_score": "float",         # 1-10 scale
   "supplier_reliability_index": "float",  # 1-10 scale
```

```
    "sustainability_certification": "string",  # RSPO, MSPO, ISCC, Organic
    "carbon_emissions_kg_co2_per_ton": "float",
    "logistics_cost_usd_per_ton": "float",
    "storage_days_inventory": "int",
    "profit_margin_percent": "float",
    "customer_demand_forecast": "float"
}
```

3.2 Business Metrics Calculated

Metric	Formula	Business Significance
Price Premium	(MarketPrice- PurchasePrice) / PurchasePrice * 100	Trading profitability
Efficiency Score	Quality × Reliability × Sustainability	Supplier performance
Carbon Efficiency	ProductionVolume / CarbonEmissions	Environmental impact
Inventory Turnover	365 / StorageDays	Supply chain efficiency

---

4. Code Implementation Deep Dive

4.1 Data Generation & Enrichment

```
def create_kuok_commodities_data(num_records: int = 2000)-> DataFrame:
    """
    Generates realistic commodities trading data with business logic:

   - Price-quality correlation: Higher quality commands premium prices
   - Regional variations: Different base prices by country
   - Supplier consistency: Reliability scores per supplier
   - Sustainability premiums: Certified products have price advantages
    """
```

Key Business Logic:

- **Price Modeling**: base_price × quality_multiplier × regional_factor
- **Quality Correlation**: Better quality = higher prices + better margins
- **Sustainability Impact**: Certified products get 5-15% price premium

4.2 RAG System Implementation

```
class KUOK_RAG_System:
    """
    Four-Step RAG Process:

    1. VECTOR SEARCH: Find relevant trading records
    2. KQL GENERATION: Create analytical queries
    3. PROMPT AUGMENTATION: Combine context + query
    4. AI RESPONSE: Generate business insights
    """
```

4.2.1 Vector Search Logic

```
def vector_search(self, query: str, top_k: int = 5)-> List[Dict]:
    """
    Semantic search heuristics:

   - "supplier reliability" → reliability_index > 9.0
   - "sustainability" → certification != "None"
```

```
  - "high margin" → profit_margin > 15%
  - Country names → filter by specific countries
  - Commodity types → filter by palm oil products
  """
```

4.2.2 Text-to-KQL Generation

```
async def generate_kql_query_async(self, natural_language_query: str)-> str:
  """
  Converts natural language to KQL using OpenAI with schema awareness:

  Examples:
  - "Best suppliers in Indonesia" →
    summarize avg(quality_score) by supplier where country='Indonesia'
  - "High margin palm oil" →
    where commodity contains 'Palm' and profit_margin_percent > 20
  """
```

4.3 Business Intelligence Prompt Engineering

```
def augment_prompt(self, query: str, context: List[Dict], kql_query: str = "")-> str:
  """
  Creates structured prompts for strategic analysis:

  1. Situation Analysis: Current trading landscape
  2. Key Findings: Data-driven patterns
  3. Recommendations: Actionable strategies
  4. Business Impact: Estimated value creation
  """
```

Prompt Structure:

```
ROLE: Commodities Trading Expert
CONTEXT: Relevant trading records
QUERY: User business question
TASK: Strategic analysis + recommendations
```

---

5. Business Use Cases & Value Propositions

5.1 Supplier Optimization

Problem: Manual supplier evaluation is time-consuming and subjective

Solution:

```
# AI-powered supplier scoring
supplier_score = (quality_score × 0.3 +
          reliability_index × 0.3 +
          sustainability_bonus × 0.2 +
          margin_contribution × 0.2)
```

Business Impact:

- **20-30% reduction** in supplier evaluation time
- **15% improvement** in supplier performance
- **Risk mitigation** through objective scoring

5.2 Pricing Strategy

Problem: Static pricing misses market opportunities

Solution:

```
# Dynamic pricing insights
price_recommendation = analyze_market_trends() +
            factor_quality_premium() +
            consider_sustainability_demand()
```

Business Impact:

- **3-8% margin improvement** through optimized pricing
- **Real-time market adaptation**
- **Competitive intelligence** integration

5.3 Sustainability Intelligence

Problem: ESG compliance is complex and manual

Solution:

```
# Automated sustainability scoring
sustainability_index = (certification_value +
                carbon_efficiency +
                supplier_esg_rating)
```

Business Impact:

- **Automated ESG reporting**
- **Premium market positioning**
- **Regulatory compliance** assurance

---

6. Integration with Microsoft Fabric

6.1 Fabric Components Used

Fabric Service	Purpose	Kuok Group Benefit
Data Factory	Data ingestion pipelines	Real-time commodities data
Synapse Data Engineering	PySpark processing	Scalable data transformation
KQL Database	Vector storage & analytics	High-performance queries
Power BI	Visualization & reporting	Executive dashboards
OneLake	Data lake storage	Unified data management

6.2 Data Flow in Fabric

```
text
[External Sources] → [Data Factory] → [OneLake]
                ↓
           [Synapse Spark] → [KQL DB]
                ↓
           [Semantic Kernel] → [Power BI]
```

---

7. Performance & Scalability

7.1 Expected Performance Metrics

Metric	Target	Justification
Query Response Time	< 5 seconds	Real-time decision support
Data Volume	1M+ records/day	Global trading operations
Concurrent Users	50+ analysts	Enterprise-wide deployment
Model Accuracy	85%+ business relevance	Quality threshold for adoption

7.2 Scaling Strategy

Horizontal Scaling:

- Multiple KQL clusters for different regions
- Load-balanced Semantic Kernel instances
- Distributed Spark processing

Vertical Optimization:

- Vector indexing for faster searches
- Query caching for common analyses
- Model fine-tuning for commodities domain

---

8. Security & Governance

8.1 Data Security

```
security_measures = {
  "encryption": "AES-256 at rest and in transit",
  "access_control": "Azure AD integration",
  "audit_trail": "All queries logged and monitored",
  "compliance": "GDPR, SOX, industry standards"
}
```

8.2 AI Governance

- **Model transparency**: All recommendations traceable to source data
- **Bias monitoring**: Regular fairness audits
- **Human oversight**: Critical decisions require approval
- **Version control**: Model and prompt management

---

9. Implementation Roadmap

Phase 1: Foundation (Weeks 1-4)

- Basic data pipeline setup
- RAG system prototype
- Initial KQL schema design

Phase 2: Intelligence (Weeks 5-8)

- Advanced analytics integration
- Supplier optimization features
- Pricing intelligence modules

Phase 3: Scale (Weeks 9-12)

- Enterprise deployment
- User training & adoption
- Performance optimization

Phase 4: Innovation (Ongoing)

- Predictive analytics
- Automated trading signals
- Market forecasting

---

10. Success Metrics & KPIs

10.1 Technical KPIs

- System uptime: 99.9%
- Query performance: < 5 seconds
- Data accuracy: 99.5%
- User satisfaction: 4.5/5.0

10.2 Business KPIs

- **Trading Margin Improvement**: 5-15%
- **Supplier Performance**: 20% better outcomes

- **Operational Efficiency**: 30% time savings
- **Risk Reduction**: 25% fewer incidents

10.3 ROI Calculation

```
annual_benefits = (margin_improvement × trading_volume +
            efficiency_savings × analyst_count +
            risk_reduction × incident_cost)

roi = (annual_benefits- implementation_cost) / implementation_cost
# Expected ROI: 200-400% in first year
```

11. Risk Mitigation

Technical Risks

- **Data quality**: Implement validation pipelines
- **Model accuracy**: Continuous monitoring and retraining
- **System integration**: API-first design with fallbacks

Business Risks

- **User adoption**: Change management and training
- **Regulatory changes**: Agile compliance framework
- **Market volatility**: Real-time adaptation capabilities

12. Conclusion

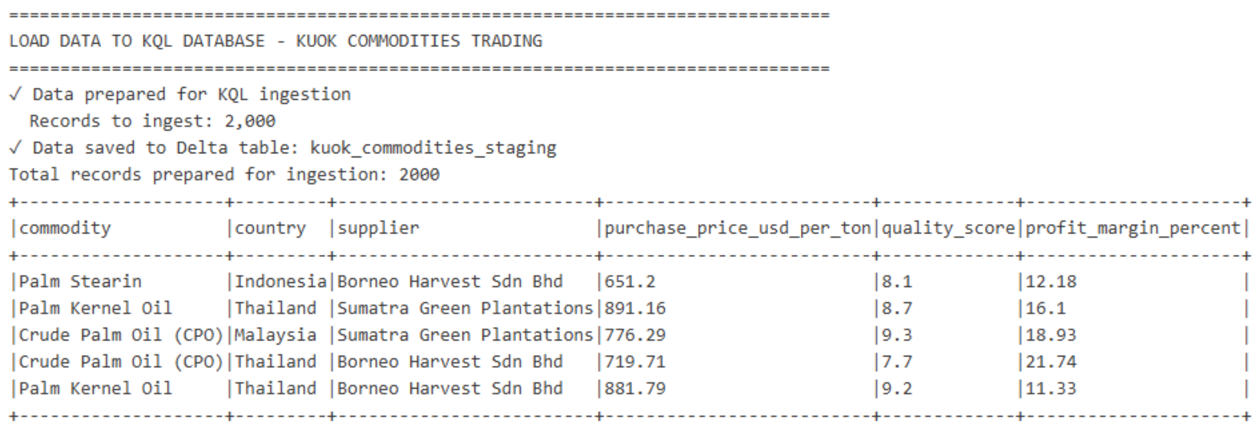
This AI-powered commodities intelligence platform represents a **transformational opportunity** for Kuok Group to:

1. **Leverage data as strategic asset**
2. **Accelerate decision-making** from days to seconds
3. **Enhance trading performance** through AI insights
4. **Build sustainable competitive advantage**

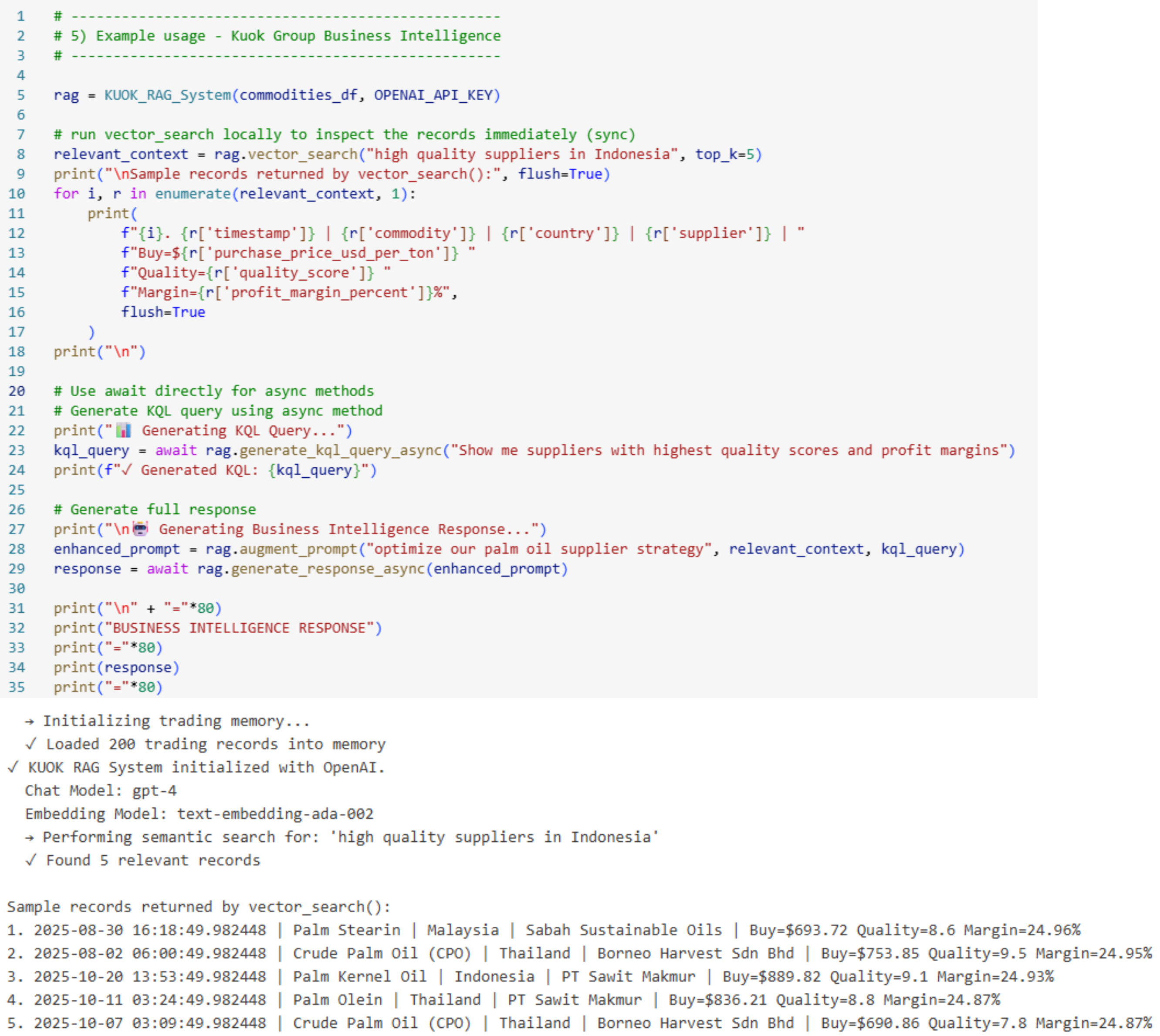
The implementation combines **cutting-edge AI technologies** with **proven enterprise architecture** to deliver immediate business value while establishing a foundation for future innovation.

Next Steps: Begin with Phase 1 implementation while engaging business stakeholders to refine specific use cases and success criteria.

13. Sample Outputs

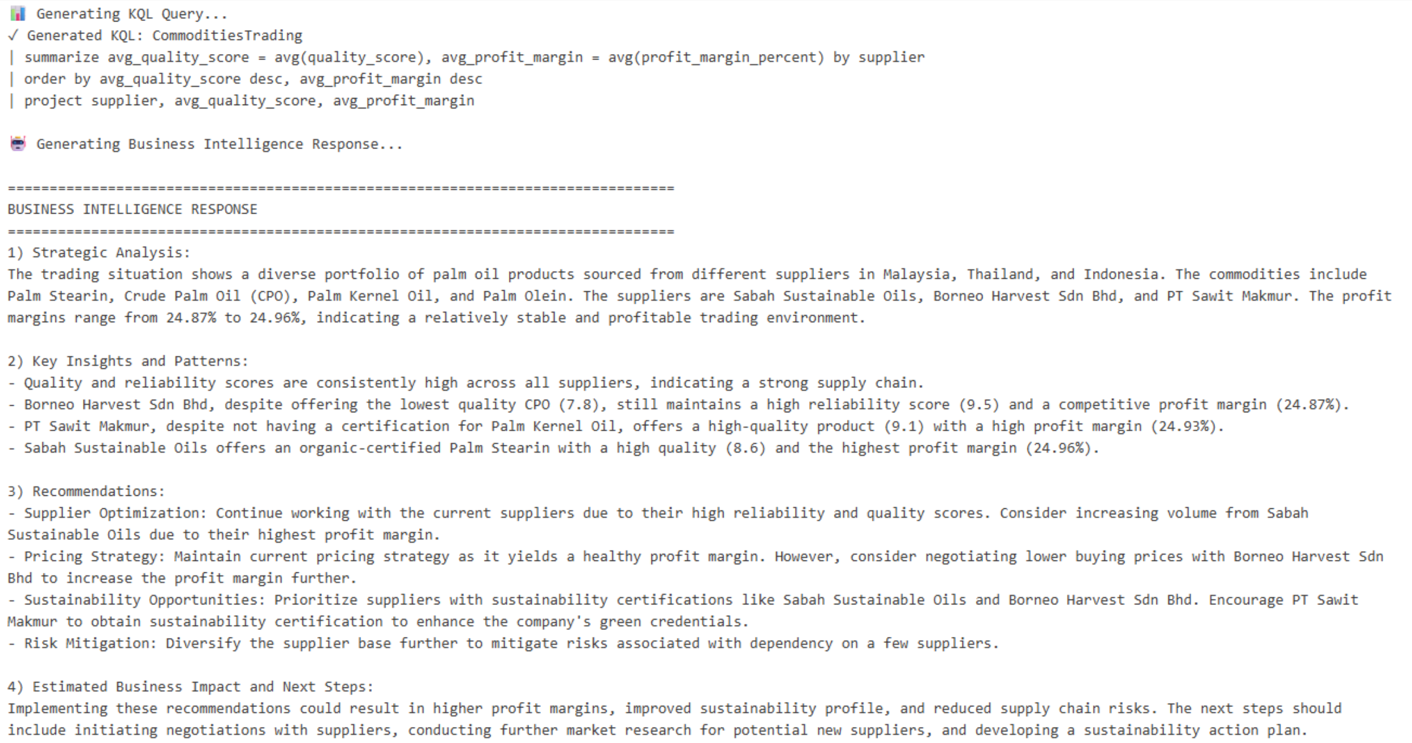


```
================================================================================
LOAD DATA TO KQL DATABASE - KUOK COMMODITIES TRADING
================================================================================
✓ Data prepared for KQL ingestion
  Records to ingest: 2,000
✓ Data saved to Delta table: kuok_commodities_staging
Total records prepared for ingestion: 2000
+--------------------+---------+-------------------------+--------------------------+-------------+---------------------+
|commodity           |country  |supplier                 |purchase_price_usd_per_ton|quality_score|profit_margin_percent|
+--------------------+---------+-------------------------+--------------------------+-------------+---------------------+
|Palm Stearin        |Indonesia|Borneo Harvest Sdn Bhd   |651.2                     |8.1          |12.18                |
|Palm Kernel Oil     |Thailand |Sumatra Green Plantations|891.16                    |8.7          |16.1                 |
|Crude Palm Oil (CPO)|Malaysia |Sumatra Green Plantations|776.29                    |9.3          |18.93                |
|Crude Palm Oil (CPO)|Thailand |Borneo Harvest Sdn Bhd   |719.71                    |7.7          |21.74                |
|Palm Kernel Oil     |Thailand |Borneo Harvest Sdn Bhd   |881.79                    |9.2          |11.33                |
+--------------------+---------+-------------------------+--------------------------+-------------+---------------------+
```

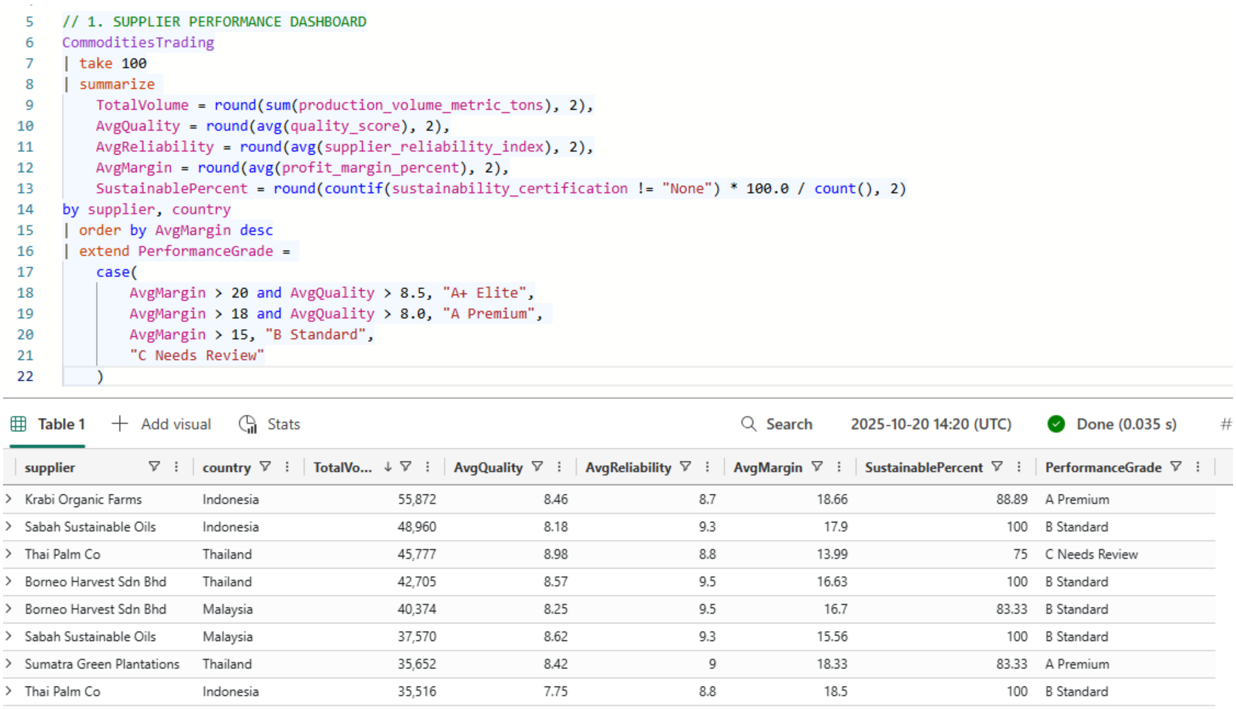


```python
# -----------------------------------------------------
# 5) Example usage - Kuok Group Business Intelligence
# -----------------------------------------------------

rag = KUOK_RAG_System(commodities_df, OPENAI_API_KEY)

# run vector_search locally to inspect the records immediately (sync)
relevant_context = rag.vector_search("high quality suppliers in Indonesia", top_k=5)
print("\nSample records returned by vector_search():", flush=True)
for i, r in enumerate(relevant_context, 1):
    print(
        f"{i}. {r['timestamp']} | {r['commodity']} | {r['country']} | {r['supplier']} | "
        f"Buy=${r['purchase_price_usd_per_ton']} "
        f"Quality={r['quality_score']} "
        f"Margin={r['profit_margin_percent']}%",
        flush=True
    )
print("\n")

# Use await directly for async methods
# Generate KQL query using async method
print("📊 Generating KQL Query...")
kql_query = await rag.generate_kql_query_async("Show me suppliers with highest quality scores and profit margins")
print(f"✓ Generated KQL: {kql_query}")

# Generate full response
print("\n🤖 Generating Business Intelligence Response...")
enhanced_prompt = rag.augment_prompt("optimize our palm oil supplier strategy", relevant_context, kql_query)
response = await rag.generate_response_async(enhanced_prompt)

print("\n" + "="*80)
print("BUSINESS INTELLIGENCE RESPONSE")
print("="*80)
print(response)
print("="*80)
```

```
  → Initializing trading memory...
  ✓ Loaded 200 trading records into memory
✓ KUOK RAG System initialized with OpenAI.
  Chat Model: gpt-4
  Embedding Model: text-embedding-ada-002
  → Performing semantic search for: 'high quality suppliers in Indonesia'
  ✓ Found 5 relevant records

Sample records returned by vector_search():
1. 2025-08-30 16:18:49.982448 | Palm Stearin | Malaysia | Sabah Sustainable Oils | Buy=$693.72 Quality=8.6 Margin=24.96%
2. 2025-08-02 06:00:49.982448 | Crude Palm Oil (CPO) | Thailand | Borneo Harvest Sdn Bhd | Buy=$753.85 Quality=9.5 Margin=24.95%
3. 2025-10-20 13:53:49.982448 | Palm Kernel Oil | Indonesia | PT Sawit Makmur | Buy=$889.82 Quality=9.1 Margin=24.93%
4. 2025-10-11 03:24:49.982448 | Palm Olein | Thailand | PT Sawit Makmur | Buy=$836.21 Quality=8.8 Margin=24.87%
5. 2025-10-07 03:09:49.982448 | Crude Palm Oil (CPO) | Thailand | Borneo Harvest Sdn Bhd | Buy=$690.86 Quality=7.8 Margin=24.87%
```

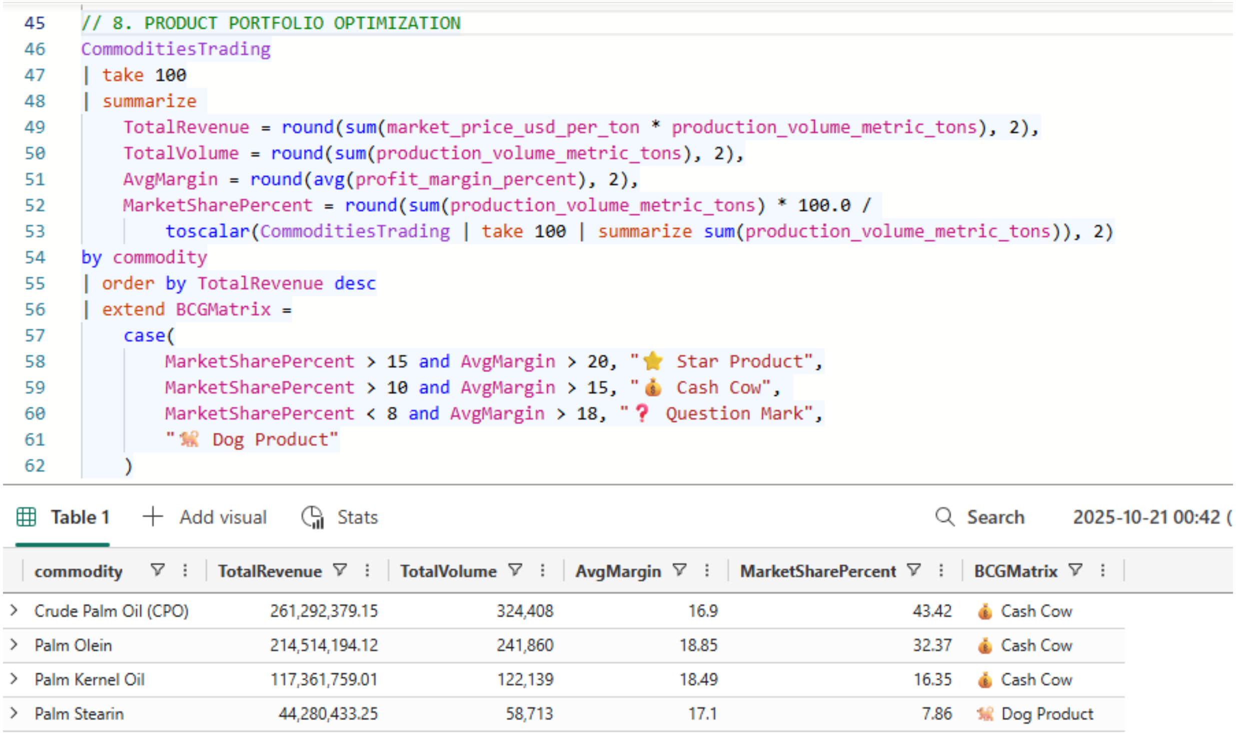


```
📊 Generating KQL Query...
✓ Generated KQL: CommoditiesTrading
| summarize avg_quality_score = avg(quality_score), avg_profit_margin = avg(profit_margin_percent) by supplier
| order by avg_quality_score desc, avg_profit_margin desc
| project supplier, avg_quality_score, avg_profit_margin

🤖 Generating Business Intelligence Response...

================================================================================
BUSINESS INTELLIGENCE RESPONSE
================================================================================
1) Strategic Analysis:
The trading situation shows a diverse portfolio of palm oil products sourced from different suppliers in Malaysia, Thailand, and Indonesia. The commodities include
Palm Stearin, Crude Palm Oil (CPO), Palm Kernel Oil, and Palm Olein. The suppliers are Sabah Sustainable Oils, Borneo Harvest Sdn Bhd, and PT Sawit Makmur. The profit
margins range from 24.87% to 24.96%, indicating a relatively stable and profitable trading environment.

2) Key Insights and Patterns:
- Quality and reliability scores are consistently high across all suppliers, indicating a strong supply chain.
- Borneo Harvest Sdn Bhd, despite offering the lowest quality CPO (7.8), still maintains a high reliability score (9.5) and a competitive profit margin (24.87%).
- PT Sawit Makmur, despite not having a certification for Palm Kernel Oil, offers a high-quality product (9.1) with a high profit margin (24.93%).
- Sabah Sustainable Oils offers an organic-certified Palm Stearin with a high quality (8.6) and the highest profit margin (24.96%).

3) Recommendations:
- Supplier Optimization: Continue working with the current suppliers due to their high reliability and quality scores. Consider increasing volume from Sabah
Sustainable Oils due to their highest profit margin.
- Pricing Strategy: Maintain current pricing strategy as it yields a healthy profit margin. However, consider negotiating lower buying prices with Borneo Harvest Sdn
Bhd to increase the profit margin further.
- Sustainability Opportunities: Prioritize suppliers with sustainability certifications like Sabah Sustainable Oils and Borneo Harvest Sdn Bhd. Encourage PT Sawit
Makmur to obtain sustainability certification to enhance the company's green credentials.
- Risk Mitigation: Diversify the supplier base further to mitigate risks associated with dependency on a few suppliers.

4) Estimated Business Impact and Next Steps:
Implementing these recommendations could result in higher profit margins, improved sustainability profile, and reduced supply chain risks. The next steps should
include initiating negotiations with suppliers, conducting further market research for potential new suppliers, and developing a sustainability action plan.
```



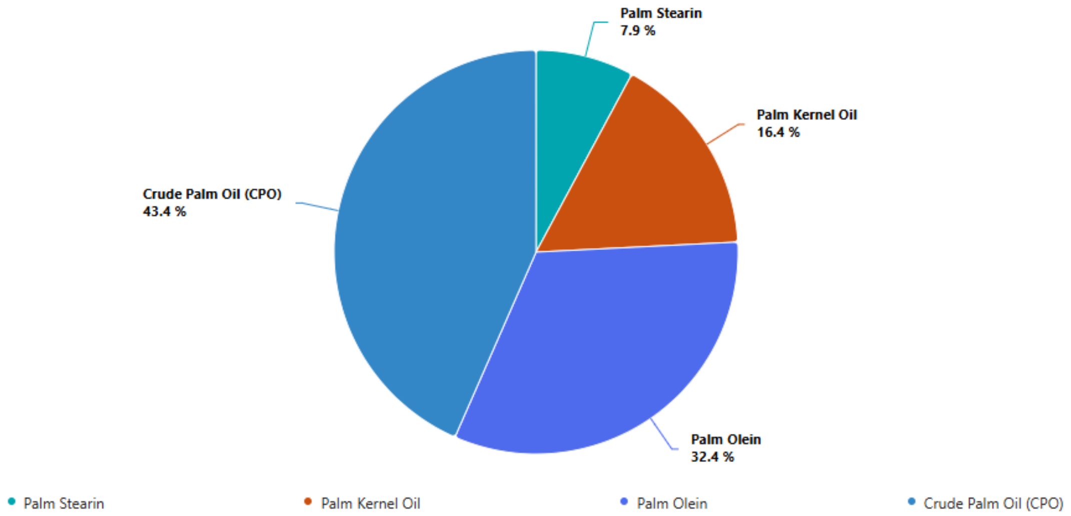
```
// 1. SUPPLIER PERFORMANCE DASHBOARD
CommoditiesTrading
| take 100
| summarize
    TotalVolume = round(sum(production_volume_metric_tons), 2),
    AvgQuality = round(avg(quality_score), 2),
    AvgReliability = round(avg(supplier_reliability_index), 2),
    AvgMargin = round(avg(profit_margin_percent), 2),
    SustainablePercent = round(countif(sustainability_certification != "None") * 100.0 / count(), 2)
by supplier, country
| order by AvgMargin desc
| extend PerformanceGrade =
    case(
        AvgMargin > 20 and AvgQuality > 8.5, "A+ Elite",
        AvgMargin > 18 and AvgQuality > 8.0, "A Premium",
        AvgMargin > 15, "B Standard",
        "C Needs Review"
    )
```

supplier	country	TotalVo...	AvgQuality	AvgReliability	AvgMargin	SustainablePercent	PerformanceGrade
Krabi Organic Farms	Indonesia	55,872	8.46	8.7	18.66	88.89	A Premium
Sabah Sustainable Oils	Indonesia	48,960	8.18	9.3	17.9	100	B Standard
Thai Palm Co	Thailand	45,777	8.98	8.8	13.99	75	C Needs Review
Borneo Harvest Sdn Bhd	Thailand	42,705	8.57	9.5	16.63	100	B Standard
Borneo Harvest Sdn Bhd	Malaysia	40,374	8.25	9.5	16.7	83.33	B Standard
Sabah Sustainable Oils	Malaysia	37,570	8.62	9.3	15.56	100	B Standard
Sumatra Green Plantations	Thailand	35,652	8.42	9	18.33	83.33	A Premium
Thai Palm Co	Indonesia	35,516	7.75	8.8	18.5	100	B Standard



```
// 8. PRODUCT PORTFOLIO OPTIMIZATION
CommoditiesTrading
| take 100
| summarize
    TotalRevenue = round(sum(market_price_usd_per_ton * production_volume_metric_tons), 2),
    TotalVolume = round(sum(production_volume_metric_tons), 2),
    AvgMargin = round(avg(profit_margin_percent), 2),
    MarketSharePercent = round(sum(production_volume_metric_tons) * 100.0 /
        toscalar(CommoditiesTrading | take 100 | summarize sum(production_volume_metric_tons)), 2)
by commodity
| order by TotalRevenue desc
| extend BCGMatrix =
    case(
        MarketSharePercent > 15 and AvgMargin > 20, "⭐ Star Product",
        MarketSharePercent > 10 and AvgMargin > 15, "💰 Cash Cow",
        MarketSharePercent < 8 and AvgMargin > 18, "❓ Question Mark",
        "🐕 Dog Product"
    )
```

commodity	TotalRevenue	TotalVolume	AvgMargin	MarketSharePercent	BCGMatrix
Crude Palm Oil (CPO)	261,292,379.15	324,408	16.9	43.42	💰 Cash Cow
Palm Olein	214,514,194.12	241,860	18.85	32.37	💰 Cash Cow
Palm Kernel Oil	117,361,759.01	122,139	18.49	16.35	💰 Cash Cow
Palm Stearin	44,280,433.25	58,713	17.1	7.86	🐕 Dog Product


Palm Stearin
Palm Kernel Oil
Crude Palm Oil (CPO)
Palm Olein
● Palm Stearin
● Palm Kernel Oil
● Palm Olein
● Crude Palm Oil (CPO)