Capstone Project 1: Enterprise Lakehouse QE Pipeline with Native Validation & SLA Enforcement

Covers Modules: 1, 2, 3, 4

Business Scenario:
A retail organization wants to build an **end-to-end QE framework** for their sales and inventory data. They need:

- Bronze → Silver → Gold pipelines
- **Delta Live Tables expectations** to validate data at every stage
- **SLA enforcement** to ensure Gold KPIs are updated on time
- **Anomaly detection** using AI
- **Lineage, governance, and scorecards** for executives

Architecture Diagram (Conceptual):

```
[Raw Data Ingest (S3/ADLS)]
        |
        v
   [Bronze Layer] -- Delta Expectations --> [Silver Layer] -- SLA + AI
checks --> [Gold Layer: KPIs]
        |
   Unity Catalog: Schema + Lineage + Access Controls
```

Implementation Steps

Step 1: Bronze Layer with Delta Expectations

```
import dlt
from pyspark.sql.functions import col

@dlt.table
@dlt.expect("valid_schema", "amount IS NOT NULL")
@dlt.expect("rescued_data_clean", "_rescued_data IS NULL")
def bronze_sales():
    return spark.readStream.format("cloudFiles")\
        .option("cloudFiles.format", "json")\
        .load("/mnt/sales_data")
```

Step 2: Silver Layer with Fail-Gates

```
@dlt.table
@dlt.expect_all_or_fail({
    "positive_amount": "amount > 0",
    "valid_product": "product_id IS NOT NULL"
})
def silver_sales():
    return dlt.read("bronze_sales")
```

Step 3: Gold Layer & SLA Enforcement

```
from pyspark.sql.functions import sum, to_date, current_timestamp

@dlt.table
@dlt.expect_or_fail("daily_revenue_threshold", "total_revenue > 100000")
def gold_sales():
    return dlt.read("silver_sales")\
        .withColumn("date", to_date("txn_ts"))\
        .groupBy("date")\
        .agg(sum("amount").alias("total_revenue"))
# SLA check: Gold must be refreshed within 30 mins
gold_table = "gold_sales"
sla_minutes = 30

sla_status = spark.sql(f"""
SELECT max(load_ts) as last_load, current_timestamp() as now
FROM {gold_table}
""").collect()[0]

last_load, now = sla_status['last_load'], sla_status['now']
diff_minutes = (now - last_load).total_seconds() / 60
if diff_minutes > sla_minutes:
    raise Exception(f"SLA Breach: Gold table not refreshed in
{diff_minutes:.1f} mins")
```

Step 4: AI-based Anomaly Detection (Photon/ML)

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.clustering import KMeans

df_features = VectorAssembler(inputCols=["amount"],
outputCol="features").transform(spark.read.table("silver_sales"))
kmeans = KMeans(k=2, seed=42).fit(df_features)
pred = kmeans.transform(df_features)
pred.show(5)
```

Step 5: Governance & Scorecards

```
# Example: Lineage + Data Quality Scorecard
dlt.read("bronze_sales").display()
dlt.read("silver_sales").display()
dlt.read("gold_sales").display()
```

✅ Skills Covered:

- Enterprise Bronze → Silver → Gold QE pipelines
- Delta Expectations, SLA-driven workflows
- AI-driven anomaly detection
- Unity Catalog governance and scorecards
- Operational & executive-ready reporting

Capstone Project 2: AI-driven QE & Business Narrative with MLflow Explainability

Covers Modules: 5, 6, 7, 8

Business Scenario:
A **CPG company** wants to validate their **product master & inventory data** and ensure **model governance, drift monitoring, and explainability** to meet regulatory and executive requirements.

Key Goals:

- Track feature drift and data freshness
- Generate **business-friendly SHAP explanations**
- Detect anomalies in product and inventory data
- Use AI Agents for automated root cause analysis
- Prepare **RFP-ready architecture and KPI narrative**

Implementation Steps

Step 1: Track Feature Drift (MLflow)

```
import mlflow
from mlflow.models.signature import infer_signature

# Log features and model
model = rf_model  # pre-trained RandomForest
mlflow.sklearn.log_model(model, "rf_model",
signature=infer_signature(X_train, y_train))

# Track feature drift
from evidently.dashboard import Dashboard
from evidently.dashboard.tabs import DataDriftTab

data_drift_dashboard = Dashboard(tabs=[DataDriftTab()])
data_drift_dashboard.calculate(X_ref=X_train, X_current=X_test)
data_drift_dashboard.show()
```

Step 2: Explainability for Business

```
import shap

explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)

# Top 3 features for first customer
idx = 0
```

```
top_features = pd.DataFrame({
    'feature': feature_cols,
    'shap_value': shap_values[1][idx]  # Positive class
}).sort_values('shap_value', key=abs, ascending=False).head(3)

for i, row in top_features.iterrows():
    impact = "increases" if row.shap_value > 0 else "decreases"
    print(f"{row.feature} {impact} risk (SHAP: {row.shap_value:.4f})")

# SHAP summary plot
shap.summary_plot(shap_values, X_test, feature_names=feature_cols)
```

Step 3: AI Agents for Root Cause Analysis

```
# Example: Detect anomalies in inventory
from pyspark.sql.functions import col

inventory_df = spark.read.table("silver_inventory")
anomalies = inventory_df.filter((col("stock_level") < 0) |
(col("last_updated").isNull()))
anomalies.display()
```

Step 4: RFP-ready Architecture & Narrative

Deliverables:

- Architecture diagram showing **Bronze → Silver → Gold**, **AI QE layer**, **Governance & MLflow**
- KPI narrative: Faster insights, reduced escalations
- Cost savings comparison: Native DLT vs external DQ tools
- Next steps for certification & enterprise rollout

✅ **Skills Covered:**

- AI Agents for QE
- MLflow governance & feature drift monitoring
- SHAP explainability for business & regulatory teams
- Retail/CPG quality validation
- Executive-ready RFP narrative

Comparison Table of Capstones

Capstone	Modules Covered	Business Focus	Key Skills
1: Enterprise QE Pipeline	1,2,3,4	Retail sales & inventory pipelines	Bronze → Silver → Gold, Delta Expectations, SLA, AI anomaly detection, governance

Capstone	Modules Covered	Business Focus	Key Skills
2: AI-driven QE & Business Narrative	5,6,7,8	Product master & inventory validation	Feature drift, SHAP explainability, AI Agents, MLflow, RFP narrative

---

These two capstone projects together give a **full enterprise QE lifecycle**:

- **Project 1:** Data engineering, QE, and automation
- **Project 2:** AI governance, model explainability, and business storytelling

---