GlobalMart: Real-Time E-Commerce Analytics Platform

Duration	Team Size	Total Points
4-5 weeks	3 students	100 (20% of final grade)

Project Overview

Design and implement a real-time analytics platform for an e-commerce company that processes streaming transaction data, performs batch analytics, and provides insights through dashboards and APIs.

Business Scenario

You are hired by "GlobalMart," an e-commerce platform with:

- 10 million active users
- 250,000 products across 100 categories
- Operations in 5 countries
- 100,000 daily transactions
- Need for real-time monitoring and analytics
- Business intelligence dashboards for management

Project Components

1. Data Generation and Ingestion (20 points)

Requirements:

- Generate realistic e-commerce data streams including user transactions, product views, and cart events
- Implement Kafka producers for different data streams
- Data quality validation and duplicate handling
- Target throughput: 500 events per second

Deliverables:

- Data generator script with configurable parameters
- Kafka topic configuration and producers
- Basic monitoring dashboard showing ingestion rates

2. Stream Processing Pipeline (30 points)

Requirements:

- Real-time transaction monitoring with basic anomaly detection
- Real-time inventory tracking and low-stock alerts
- Session analysis and shopping cart abandonment detection
- Real-time sales metrics aggregation (by hour, category, region)

Deliverables:

- Spark Streaming or Flink application
- Alert notification system for critical events
- Real-time metrics dashboard

3. Batch Processing and Data Warehouse (25 points)

Requirements:

- Daily batch jobs for customer segmentation (RFM analysis)
- Product performance analysis
- Sales trend analysis
- Design a simple star schema data warehouse

Deliverables:

- ETL pipeline using Spark
- Data warehouse schema documentation
- Scheduled batch job scripts

4. Visualization and API Layer (15 points)

Requirements:

- Executive dashboard showing key metrics
- RESTful API endpoints for data access
- Geographic sales distribution visualization
- Product category performance charts

Deliverables:

- Interactive dashboard (Grafana, Tableau, or Power BI)
- API documentation with example queries
- Web interface for data exploration

5. Infrastructure and Deployment (10 points)

Requirements:

- Working deployment using Docker containers or local cluster
- System monitoring and health checks
- Basic security measures (authentication, data validation)

Deliverables:

- Deployment scripts and configuration files
- System architecture diagram
- Monitoring dashboard for system health

Technical Stack

Required Technologies:

- Storage: HDFS, S3, or local filesystem
- Streaming: Kafka + (Spark Streaming or Flink)
- Batch Processing: Apache Spark
- Database: Choose 1-2 from (PostgreSQL, MongoDB, Cassandra)
- Visualization: Grafana, Tableau, or Power BI

Optional Technologies:

- Docker for containerization
- Redis for caching
- Prometheus for system monitoring

Final Deliverables

1. Code Repository

- Well-organized GitHub repository with clear structure
- Comprehensive README with setup instructions
- Code comments and inline documentation
- Requirements file and dependency management

2. Technical Documentation

Architecture Document:

- System architecture diagram
- Data flow diagrams
- Technology selection justification
- Design decisions and trade-offs

User Guide:

- Setup and installation instructions
- Configuration guide
- Troubleshooting common issues
- API documentation

3. Live Demonstration

15-20 minute presentation covering:

- Business problem overview
- Architecture walkthrough
- Live demonstration of all system components
- Performance metrics and system behavior
- Challenges faced and solutions implemented
- Team collaboration and individual contributions

4. Individual Contribution Report

Each team member submits (2-3 pages):

- Personal contributions to the project
- Technical challenges and how they were solved
- Key learnings and skills developed
- Peer evaluation of team members

Quality Assessment Criteria

Technical Excellence (40%):

- Code quality, organization, and best practices
- Architecture design and scalability considerations
- Error handling and system reliability
- Performance optimization

Implementation Completeness (30%):

- All required components functioning correctly
- Integration between system components
- Demonstration of working end-to-end pipeline

Documentation Quality (20%):

- Clarity and completeness of technical documentation
- Quality of architecture diagrams and explanations
- README and setup instructions
- Code comments and API documentation

Teamwork and Presentation (10%):

- Git commit history and contribution balance
- Quality of live demonstration
- Peer evaluations
- Professional presentation skills

Recommended Reading:

- "Designing Data-Intensive Applications" by Martin Kleppmann
- "Spark: The Definitive Guide" by Chambers and Zaharia
- Apache Kafka and Apache Flink documentation

Sample Data Schemas

User Profile Schema

```
{
      "user_id": "string",
      "email": "string",
      "age": integer,
      "country": "string",
      "registration_date": "timestamp",
      "preferences": ["array of categories"]
}
```

Transaction Event Schema

```
{
      "transaction_id": "string",
      "user_id": "string",
      "timestamp": "timestamp",
      "products": [{
            "product_id": "string",
            "quantity": integer,
            "price": float
      }],
      "total_amount": float,
      "payment_method": "string"
}
```

Product Catalog Schema

```
{
      "product_id": "string",
      "name": "string",
      "category": "string",
      "price": float,
      "inventory": integer,
      "ratings": float
}
```

Tips for Success

1. **Start with MVP:** Get a simple end-to-end pipeline working before adding advanced features
2. **Divide Work Clearly:** Assign specific components to team members based on their strengths
3. **Use Version Control:** Commit frequently with descriptive messages
4. **Test Components Separately:** Verify each component works independently before integration
5. **Document as You Build:** Don't leave all documentation for the end
6. **Plan for Integration Time:** Budget extra time for connecting all components
7. **Practice Your Demo:** Rehearse the presentation multiple times before the final demo

Academic Integrity

- All code must be original work or properly attributed
- External libraries and frameworks are allowed with proper documentation
- Teams must maintain a clear contribution log in Git
- Plagiarism will result in project failure and potential course consequences
- Collaboration between teams is not permitted

This project provides hands-on experience with real-world big data systems and prepares you for careers in data engineering and analytics.