**EQREV Hackathon: Agentic AI for Quick Commerce**

# Problem Statement

Quick commerce companies deliver goods to consumers within minutes. Efficient operations require:  
• Allocating inventory wisely across cities.  
• Predicting demand.  
• Fulfilling timely of stock (restocking) to quick commerce companies.  
• Automating operational tasks like sending supplier emails or alerts.  
  
Your challenge is to build an Agentic AI System that acts like a virtual operations manager.  
  
This is not just a chatbot that answers queries — your agent should:  
1. Analyze data (sales, inventory, city performance).  
2. Generate actionable insights (e.g., “Increase PO for City A”, “Move 300 units to City B”).  
3. Take autonomous actions (send email alerts, trigger supplier notifications, update dashboards).

# Expected Agent Behaviors

## 1. Decision-Making

• Understand queries like:  
 o “If I have 1000 units of Product X, where should I place them?”  
 o “Which cities are underperforming this week?”  
• Query structured data (Postgres DB / CSV).  
• Run analytics or sales forecasting (using Pandas/Polars, scikit-learn/Prophet).  
• Output actionable decisions, not vague answers.

## 2. Action Execution (via n8n Automation)

• Instead of writing custom automation, students must integrate with n8n to trigger workflows.  
  
Example Flows:  
• Email Automation → n8n workflow:  
 - Sends a notification to warehouse to send more stocks to quick commerce companies if stock < sales threshold.  
 - If stock availability itself is less than required units, trigger notification to supplier for manufacturing new units (purchase order).  
• Database/API Update → n8n workflow updates stock allocation or posts data to an external system.  
  
The agent should only generate decisions and call n8n webhooks (via HTTP requests), while n8n handles the automation flows (emails, DB updates, alerts, etc.).

## 3. Insight Generation

• Provide visualizations (charts/tables) for clarity.  
• Highlight trends (e.g., “City B sales grew 20% this week”).

# Example Scenarios to Test Agents

## Scenario 1

• User: “Allocate 1000 units of Product Y.”  
• Agent:  
 - Checks past 7-day sales → City A (40%), City B (35%), City C (25%).  
 - Suggests: “Distribute: 400 units to City A, 350 to B, 250 to C.”  
 - Sends email to warehouse team to send stock for City A.

## Scenario 2

• User: “Which products need urgent new stock to be manufactured?”  
• Agent:  
 - Finds City D stock < 20% of 3-day average sales.  
 - Responds: “City D is at risk. Raising purchase order of 500 units.”  
 - Sends Email alert automatically.

# Recommended Tech Stack

• Backend & AI → Python + LangChain, Pandas/Polars.  
• Database → Postgres (Neon.tech free tier) or DuckDB.  
• Frontend → Streamlit / Gradio.  
• Deployment → Hugging Face Spaces (free hosting).  
• Automation → Gmail SMTP / SendGrid (emails).  
• Optional Enhancements → Multi-agent setup, dashboards, real-time streaming.

# 

# Evaluation Criteria

|  |  |  |
| --- | --- | --- |
| Category | Weightage | Expectations |
| Core Functionality | 30% | Agent can analyze data and return meaningful insights with reasoning. |
| Agentic Behavior | 25% | Goes beyond answering queries — makes decisions and executes actions. |
| Data Handling & Analytics | 20% | Efficient use of DB queries, forecasting, inventory allocation logic. |
| User Experience | 15% | Smooth, interactive UI (Streamlit/Gradio app). |
| Innovation & Bonus Features | 10% | Multi-agent collaboration, dashboards, automation, or unique ideas. |

# Deliverables

1. Deployed App Link (Any Platforms) Example: Hugging Face Spaces / Streamlit Cloud / Render).  
2. GitHub Repo (code + README with setup instructions).  
3. Presentation (15 min) – Explaining architecture, demo, and insights.

# Deployment Requirement (can use any platform to deploy the complete application)

• Teams must deploy their solution so judges can test live.  
• Suggested free platforms:  
 o Hugging Face Spaces (Streamlit/Gradio/FastAPI).  
 o Streamlit Cloud.  
 o Neon.tech, Supabase (free Postgres DB).  
  
**Scoring Note:** Teams that only run locally but don’t deploy will lose points in User Experience.

# Data Usage Rules

• Complete business data should not be exposed directly to the large LLM models.  
• Instead, participants can provide sample data along with table schema to LLMs for learning purposes.  
• The agent should be designed to query only the allowed schema, ensuring security and abstraction.  
• Teams may simulate the database with Postgres, DuckDB, or CSV files but must not hardcode insights.  
• This ensures the agent reasons and acts responsibly on structured data.

**Sales Data:**

<https://docs.google.com/spreadsheets/d/1rfFmpCVXs8N9Uc67pTEtYXP5OF0bc8qLhBiOLyUXGGA/edit?usp=sharing>

**Inventory Data:**

<https://docs.google.com/spreadsheets/d/1kkHoYSV4dHmS2YOz2ki8KG0BJxP0Ywy48mrKXoTwSP8/edit?usp=sharing>