Project Statement for Milestone 1

Team Baddie

(Brian Leung, Cayden Calo, Jeremiah Carlo Miguel, Travis Takushi)

1.a.

Problem Definition:

The project implements a co-purchasing data analytics engine using NoSQL databases, Hadoop MapReduce, and/or Apache Spark MLlib. The engine will support:

- Complex Query Answering:
    - Support SQL-like queries of the form SELECT * FROM U WHERE CONDITION, where CONDITION may involve:
        - Searchable attributes: constraints over attributes in the dataset (e.g., "select products in Books group with sales rank < 500").
        - Non-searchable attributes: derived constraints such as "number of reviews > 100" or "number of co-purchased items > 5."
        - Enriched operators: queries with inequalities (>, >=, <, <=) such as "select DVDs with average rating >= 4.5."
    - The system should return the top-k entities satisfying Q with minimized query cost.
    - Implementation will be non-SQL based (leveraging distributed frameworks).

- Frequent Co-Purchasing Pattern Mining:
    - Split the dataset into training and testing subsets.
    - Identify frequent co-purchasing patterns (e.g., “Customers who buy Product A often also buy Product B and C”).
    - Verify patterns in both datasets and report their frequency.
    - Identify and output customers captured by the strongest patterns.
    - Summarize the most significant co-purchasing pattern in the dataset.

1.b.

Importance and Applications

Importance of the Problem:

Co-purchasing analysis plays a critical role in recommendation systems and targeted marketing. Online marketplaces like Amazon rely heavily on analyzing customer purchase behaviors to improve user engagement and increase sales. This dataset provides a real-world large-scale benchmark for testing distributed systems and machine learning algorithms.

Applications:

- E-commerce Recommendation Engines:
    - "Customers who bought this also bought…" suggestions.
    - Personalized product recommendations.

- Market Basket Analysis:
    - Identifying commonly purchased product bundles.
    - Cross-selling and upselling strategies.

- Business Intelligence:
    - Understanding consumer behavior.
    - Designing promotional campaigns.

- Research Applications:
    - Benchmark for evaluating distributed query processing and frequent pattern mining algorithms.

1.c.

Project Goal

Goal:

To build an end-to-end data analytics engine for Amazon's co-purchasing dataset that can:

- Efficiently process complex non-SQL queries using distributed systems (NoSQL + Spark/MapReduce).
- Discover and validate co-purchasing patterns using frequent itemset mining techniques (e.g., Apriori, FP-Growth in Spark MLlib).

Deliverables:

- A working pipeline for grabbing and searching the Amazon dataset.
- A module that flags the strongest co-purchase patterns.
- An evaluation paper that compares the different methods, so either we check MapReduce vs SparkMLlib, or SQL-like queries that flex with NoSQL on the back
- (Optional) A basic graphical interface that lets users run queries and see the results (like co-purchase graphs, circles, and links

Final Output:

The outcome is either: a full application with a simpler GUI that lets non-technical users run it themselves, or a paper that digs through the algorithm and performance trade-offs in distributed data analytics.

---

2. a.

Brian Leung: The most experience I have ever had with databases is using MongoDB for some of my projects. Otherwise, I don't have much experience working with data.

Cayden Calo: I have little to no experience in working with databases.

Jeremiah Carlo Miguel: I've worked with PostgreSQL in a few projects, most notably for my Capstone. I've also worked with MongoDB, NoSQL, and SQLite in other smaller projects. Additionally, I've used the ELK Stack (Elasticsearch, Logstash, Kibana) during my internship at 2C2P for large-scale data analytics, and I'm currently developing a payment gateway for banks in Thailand.

Travis Takushi: I have had experience working with PostgreSQL databases in our cpts 322 project, which was mainly for storing student data, including their classes that they've taken, as well as the classes they are currently taking.

---

2. b.

Our team consists of four members, and we will share the workload evenly across all tasks in this project. Each member will participate in understanding the dataset, evaluating potential data models, and researching suitable big data tools such as Spark, Hadoop, and Neo4j. We will work collaboratively to download and explore the dataset, document our findings, and prepare the milestone report together. Decisions on data modeling and tool selection will be made as a team, ensuring that all members develop a common understanding of the project requirements and contribute equally to each stage.

---

3. a.1.

The Amazon co-purchasing metadata can be represented using a key-value model, where each of the 548,552 products serves as a unique key and its associated metadata forms the value. The keys would typically be product identifiers such as ASINs (strings), while the values would be complex, JSON-like objects containing attributes such as product titles, sales ranks, category hierarchies, and customer review information including ratings, votes, helpfulness scores, and timestamps. While this approach is technically feasible, reducing the dataset to a collection of independent key-value pairs causes the relational structure—specifically the 1,788,725 co-purchase links between products—to be embedded inside the value objects rather than treated as first-class entities, which significantly diminishes the usefulness of the dataset for analyzing relationships.

3.a.2.

The Amazon co-purchasing metadata can be naturally represented as a graph, where the 548,552 unique products form the nodes and the 1,788,725 co-purchase connections between them form the edges. Each node carries descriptive attributes such as product title, ASIN, sales rank, category hierarchy, and customer review information, while edges can store relationship descriptors like the type of co-purchasing link or frequency of occurrence. The resulting graph is both labeled, since nodes and edges can be categorized with meaningful identifiers, and directed, as co-purchasing relationships may not always be symmetric. This structure highlights the interconnected nature of the dataset and supports efficient graph-based analysis.

3.a.3.

The Amazon co-purchasing metadata can also be represented in a document-oriented model, where each of the 548,552 products is stored as a separate document. Each document would contain fields such as product title, ASIN, sales rank, category hierarchy, and nested sub-elements for customer reviews, including reviewer ID, rating, vote counts, helpfulness scores, and timestamps. Co-purchasing links could either be represented as embedded lists of related products or as references to other product documents. While this model preserves much of the product-level information and supports flexible queries, the relational structure of the 1,788,725 co-purchase connections becomes less direct to navigate compared to a graph model, as traversing from one product to another through multiple levels of relationships would require repeated lookups or joins.

3.a.4.

The dataset could also be represented in a column-family data model, such as those used in wide-column stores like Cassandra. Each product could be a row identified by its ASIN, with columns for attributes like title, sales rank, categories, and reviews, while related products could be stored in additional wide columns or column families. This design would allow efficient storage and retrieval of product-level information and scale well for high-volume access patterns. However, like the key-value and document approaches, the column-family model treats co-purchasing relationships as secondary attributes rather than first-class entities, limiting its ability to efficiently support graph-based queries such as neighborhood exploration, recommendation generation, or community detection.

---

3. b.

Our team chose to represent the Amazon co-purchasing metadata as a graph because it naturally reflects both the entities (products) and their relationships (co-purchasing links) in a clear and intuitive way. In this model, the 548,552 products are represented as nodes and the 1,788,725 co-purchase connections become edges, which can be directed or labeled to capture details such as “frequently bought together.” Each node and edge can also store rich attributes, including product titles, sales ranks, category hierarchies, and review data such as timestamps, customer IDs, ratings, votes, and helpfulness scores. Using a graph structure makes these connections first-class elements rather than burying them inside nested fields as they would be in key-value or document-oriented models. This allows for efficient traversal and analysis of relationships, supporting graph-based algorithms like similarity scoring, recommendation generation, and community detection. Because the dataset’s real value lies in these interconnections, the graph model is especially well-suited for modeling and analyzing the relational complexity of the Amazon data.

4.a. Neo4j

4.b. Apache Spark with graph frame

4.c. AWS