

Assignment Title

Cassandra vs MySQL: Event Tracking System for QuickKart

1. Case Study

QuickKart is an online shopping platform where users can:

- Open the app or website
- View products
- Add products to cart
- Place an order
- Log out

Every action is recorded as an event.

QuickKart wants to:

1. Store these events in a Relational Database (MySQL)
2. Store the same events in a Wide-Column Database (Cassandra)
3. Compare how both systems handle this type of data

You are part of the data team. Your job is to design, store, and compare.

2. Learning Goals

By doing this assignment, you should be able to:

- Understand how the same data looks in MySQL and Cassandra.
- Design basic tables in both systems.
- Perform simple Create, Read, Update, Delete (CRUD) scenarios.
- Explain in your own words when MySQL is better and when Cassandra is better.

3. Synthetic Dataset (Students Will Create)

You will create fake data (synthetic data). No real users.

3.1 Minimum Dataset Size

Create at least:

- **100 users**

- **200–300 products**
- **3,000–5,000 events** in total

(If you want, you can create more events for better comparison.)

3.2 Required Fields for Each Event

Every event record should include:

1. event_id – unique ID for each event
2. user_id – which user did the action
3. session_id – one session of continuous activity
4. event_time – date and time of the event
5. event_type – choose from:
 - open_app, view_product, add_to_cart, purchase, logout
6. product_id – which product (for view/add_to_cart/purchase)
7. category – e.g., Grocery, Electronics, Clothing
8. price – price of the product (if relevant)
9. city – e.g., Lahore, Karachi, Islamabad
10. device_type – web, android, ios

Rules (keep logic simple):

- A purchase should only appear after at least one add_to_cart in the same session.
- open_app usually comes before other actions in a session.
- Some users will only browse (no purchase).
- Some users will purchase multiple times.

You can generate this data using Excel, Python, or manually.
Just keep it consistent and realistic.

4. Data Modeling Requirements

You will use the **same dataset** in two ways:

4.1 MySQL (Relational Model)

- Design a simple table (or tables) to store the events.
- Make sure you:
 - Have a primary key.
 - Can find events for a specific user.
 - Can check which products are purchased.

(Students must write their own table design and queries separately.)

4.2 Cassandra (Wide-Column Model)

- Design a table for Cassandra that is good for:
 - Quickly showing the recent events of one user.
- Choose:
 - A suitable partition key (for example, user_id).
 - A suitable way to order events (for example, by time).

(Students will write their own schema and Python + CQL code separately.)

5. CRUD Scenarios (In Words Only)

For each scenario below:

- Do it once in MySQL
- Do it once in Cassandra
- Then, briefly write:
“Which was easier? Any limitation?”

5.1 CREATE Scenarios

1. **Start a New Session**
 - A user opens the app.
 - Record a new open_app event with user, time, city, and device.
2. **User Browses Products**
 - For a given user and session, record a few view_product events for different products.
3. **User Places an Order**
 - After viewing and adding products to cart, record a purchase event for one product.

5.2 READ Scenarios

1. **User History**
 - Show the latest 10 events of a specific user.
 - Used to see what the user did recently.
2. **Product Popularity**
 - Find how many times a specific product was purchased.
 - Used to check if one product is selling well.
3. **City-wise Purchases**

- For a given day, find how many purchases happened in each city.
- Used to see which city has more orders.

5.3 UPDATE Scenarios

1. **Fix a Wrong City or Price**
 - A few events were saved with the wrong city or price.
 - Correct those values.
2. **Tag a Session as Problematic**
 - Mark one session as “problem session” (e.g., due to a bug or suspicious behavior).
 - Update all events of that session with a flag like `is_flagged`.
3. **Add a New Column Later**
 - Add a new field like `payment_method` (e.g., card, cash)
 - For some purchase events, fill this new information.

5.4 DELETE Scenarios

1. **Delete a Wrong Event**
 - Remove a single event that was added by mistake.
2. **Delete Data for One User**
 - Remove all events for a user who requested account deletion.
3. **Old Data Cleanup**
 - Assume events older than 6 months are no longer needed.
 - Remove those old events (or plan how they would be removed / expired).

6. What Students Must Submit

Each student/group must submit:

1. **Dataset Description (Short, Clear)**
 - How many users, products, events?
 - How did you generate the data?
 - 1–2 small example rows (as illustration only).
2. **MySQL Design**
 - Table design (fields + keys).
 - Written explanation of how CRUD scenarios were done.
3. **Cassandra Design**
 - Table design (keyspace, main table, partition key choice).
 - Written explanation of how CRUD scenarios were done using Python + CQL.
4. **Comparison (Simple English, 1–2 pages)**

- Which was easier for:
 - Insert many events?
 - Reading recent events of one user?
 - Doing analytics like “how many purchases per city”?
- What did you learn about:
 - When MySQL is better
 - When Cassandra is better