Use Case-1: Building a Cart Analysis for Myph – SQL and MongoDB Integration

Overview
Myph has launched a new range of phones. The business requires an integrated cart analysis system that uses both a **Relational (SQL)** database and **MongoDB (NoSQL)** for scalable product and cart management.

The SQL database manages structured transactional data (orders, stock, pricing), while MongoDB manages flexible, hierarchical, and analytical data such as product categories, cart structures, and user behavior analytics.

1. SQL Data Model (Transactional Layer)

```
The relational database manages atomic transactions with ACID properties.

Tables:
- Categories(category_id, category_name, parent_category_id, category_path)
- Products(product_id, category_id, product_name, description, stock_quantity, price)
- Customers(customer_id, customer_name, email)
- Cart(cart_id, customer_id, created_at, status)
- Cart_Items(cart_item_id, cart_id, product_id, quantity, price_at_time)
```

2. MongoDB Data Model (Analytical Layer)

```
MongoDB complements SQL by storing flexible and nested data for analytics and fast querying.

<b>Collections:</b>
- categories
- products
- carts

<b>Example Documents:</b>

// categories
{
  "_id": ObjectId("6714aa2f9a123"),
  "name": "Smartphones",
  "path": ["Electronics", "Phones", "Smartphones"],
  "parent": "Phones"
}

// products
{
  "_id": ObjectId("6714aa2f9a456"),
  "product_id": 1,
  "name": "Myph X1",
  "category": "Smartphones",
  "price": 899.99,
  "stock_quantity": 100
}

// carts
{
  "_id": ObjectId("6714aa2f9a789"),
  "customer": "Alice Johnson",
  "items": [
      { "product_name": "Myph X1", "quantity": 2, "price": 899.99 },
      { "product_name": "Myph X1 Pro", "quantity": 1, "price": 1099.99 }
  ],
  "status": "ACTIVE",
  "created_at": ISODate("2025-10-20T10:00:00Z")
}
```

3. SQL ↔ MongoDB Integration Logic

```
Integration between SQL and MongoDB can be achieved using ETL pipelines or middleware (e.g., Node.js

- SQL handles: Transactions, payments, inventory consistency.
- MongoDB handles: Category browsing, cart analytics, and recommendations.

<b>Data Synchronization Example:</b>
1. Product changes in SQL → Synced to MongoDB via background job.
2. Cart selections in MongoDB → Synced to SQL during checkout.
3. Category tree in MongoDB → Used to generate menu filters in UI.

<b>Sample Python ETL Sync (Pseudo-code):</b>
from sqlalchemy import create_engine
from pymongo import MongoClient

# SQL connection
sql_engine = create_engine('mysql+pymysql://user:pass@localhost/myph')
# MongoDB connection
mongo_client = MongoClient('mongodb://localhost:27017/')
mongo_db = mongo_client['myph_db']

# Sync Products
products = sql_engine.execute("SELECT * FROM Products").fetchall()
mongo_db.products.delete_many({})
mongo_db.products.insert_many([dict(row) for row in products])
```

4. MongoDB Analytical Queries (Cart Analysis)

```
// Get all products in category 'Smartphones'
db.products.find({ category: "Smartphones" }, { name: 1, price: 1 })

// Find average quantity of products in carts
db.carts.aggregate([
  { $unwind: "$items" },
  { $group: { _id: "$items.product_name", avgQuantity: { $avg: "$items.quantity" } } }
])

// Detect outlier (surplus) selections in carts
db.carts.aggregate([
  { $unwind: "$items" },
  { $group: { _id: "$items.product_name", avg: { $avg: "$items.quantity" }, std: { $stdDevPop: "$ite
  { $project: {
        product: "$_id",
        _id: 0,
        threshold: { $add: ["$avg", { $multiply: [2, "$std"] }] }
    }}
])
```

5. Recovery and Reliability

SQL Recovery: - Transaction rollback ensures data integrity. - Commit/rollback mechanisms maintain consistency. - Point-in-time recovery via transaction logs. **MongoDB Recovery:** - Replica sets ensure high availability and automatic failover. - Journaling provides durability for committed writes. - Backups and snapshots can restore full collections or databases.

6. Conclusion

The hybrid integration of SQL and MongoDB allows Myph to maintain strong consistency for transactional operations and leverage flexible document structures for analytical insights and fast category queries. Together, this design supports a robust, scalable e-commerce platform with real-time cart analytics and recovery mechanisms.