# Problem Statement 2: Implementing a Flexible Blog System with Azure Cosmos DB

Objective: Build a NoSQL database in Cosmos DB for a blog system using the Python SDK to create containers and insert data, incorporating consistency levels, backups, and other features.  
  
Scenario: Create a blog platform database to store posts, users, and sessions. Make it globally available, with auto-expiring sessions and real-time notifications on new posts.  
  
Data to Insert:  
  
Users Container:  
{ "userId": "u1", "name": "Alice", "email": "alice@example.com", "preferences": ["tech", "gaming"] }  
{ "userId": "u2", "name": "Bob", "email": "bob@example.com", "preferences": ["travel"] }  
{ "userId": "u3", "name": "Charlie", "email": "charlie@example.com", "preferences": ["food", "tech"] }  
{ "userId": "u4", "name": "Dana", "email": "dana@example.com", "preferences": ["gaming"] }  
{ "userId": "u5", "name": "Eve", "email": "eve@example.com", "preferences": ["travel", "food"] }  
  
Posts Container:  
{ "postId": "p1", "title": "Tech Tips", "content": "Azure basics", "tags": ["azure"], "userId": "u1" }  
{ "postId": "p2", "title": "Travel Stories", "content": "My trip to Paris", "tags": ["travel"], "userId": "u2" }  
{ "postId": "p3", "title": "Food Reviews", "content": "Best pizzas", "tags": ["food"], "userId": "u3" }  
{ "postId": "p4", "title": "Gaming News", "content": "New releases", "tags": ["gaming"], "userId": "u4" }  
{ "postId": "p5", "title": "Mixed Adventure", "content": "Travel and gaming", "tags": ["travel", "gaming"], "userId": "u5" }  
  
Sessions Container:  
{ "sessionId": "s1", "data": "active cart: laptop", "ttl": 3600, "userId": "u1" }  
{ "sessionId": "s2", "data": "active cart: mouse", "ttl": 3600, "userId": "u2" }  
{ "sessionId": "s3", "data": "active cart: keyboard", "ttl": 3600, "userId": "u3" }  
{ "sessionId": "s4", "data": "active cart: monitor", "ttl": 3600, "userId": "u4" }  
{ "sessionId": "s5", "data": "active cart: headphones", "ttl": 3600, "userId": "u5" }  
  
Steps to Solve:  
1. Install required libraries.  
2. Create Cosmos DB database and containers using Python SDK.  
3. Insert provided data into containers.  
4. Demonstrate consistency levels (Session, Strong, Eventual).  
5. Configure backups and test restore.  
6. Add global distribution and test replication.

## Assessment Questions:

1. Using Python SDK, write code to create the “blogdb” database and the three containers (users, posts, sessions).  
2. Insert all provided data into their respective containers using Python code.  
3. Write a query (in Python) to list all posts written by users who like “travel.”  
4. Retrieve all posts tagged with “gaming.”  
5. Query to find users who have more than one preference in their profile.  
6. Simulate deleting one post, then demonstrate restoring it from backup.  
7. Change the default consistency level from “Session” to “Strong.” Record any observed differences in read/write performance.  
8. Add a secondary region for global replication. Update a document in one region and verify the replication.  
9. Use TTL in the sessions container to automatically expire inactive sessions. Demonstrate it by querying after a time delay.  
10. Document your learnings on how Cosmos DB handles global distribution and consistency trade-offs.