2. Develop for Azure Storage -> 2.1 Develop solutions that use Azure Cosmos DB -> 2.1.1 Perform operations on containers and items by using the SDK

- What SDKs are supported for Cosmos DB operations?
- How do you create a container in Cosmos DB using the SDK?
- How do you insert or update an item?
- How do you query items using SQL syntax?
- How do you delete an item by ID?
- How do you use partition keys effectively?
- What are common consistency levels and how do you set them?
- How do you use the CosmosClient safely and efficiently?
- How do you handle pagination (continuation tokens)?
- How is exception handling and retry logic implemented?

1. What SDKs are supported for Cosmos DB operations?

.NET (Microsoft.Azure.Cosmos), Java, Python, Node.js

---

2. How do you create a container in Cosmos DB using the .NET SDK?

```
await database.CreateContainerIfNotExistsAsync("MyContainer", "/partitionKey");
```

- "/partitionKey" is required.
- Creates the container only if it doesn't exist.

---

3. How do you insert or update an item?

```
await container.UpsertItemAsync(item, new PartitionKey(item.partitionKey));
```

- UpsertItemAsync inserts or replaces item based on ID.
- Requires correct partition key.

---

4. How do you query items using SQL syntax?

```
var query = container.GetItemQueryIterator<MyItem>("SELECT * FROM c WHERE c.status = 'active'");
while (query.HasMoreResults)
{
   foreach (var item in await query.ReadNextAsync())
   {
      // Process item
   }
}
```

- Uses Cosmos SQL API.
- Handles pagination internally.

---

5. How do you delete an item by ID?

```
await container.DeleteItemAsync<MyItem>(id, new PartitionKey(partitionKey));
```

- Both ID and correct partition key are required.

---

6. How do you use partition keys effectively?

- Choose a key with high cardinality and even distribution (e.g., /userId).
- Required for most operations (read, update, delete).

7. What are common consistency levels and how do you set them?

- Levels: Strong, BoundedStaleness, Session (default), ConsistentPrefix, Eventual
- Set at CosmosClientOptions level:

```
new CosmosClient(endpoint, key, new CosmosClientOptions { ConsistencyLevel = ConsistencyLevel.Session });
```

---

8. How do you use the CosmosClient safely and efficiently?

- Reuse a single CosmosClient instance (thread-safe).
- Instantiate once at app startup (e.g., via dependency injection).

---

9. How do you handle pagination (continuation tokens)?

- Use `FeedIterator<T>` from `GetItemQueryIterator<T>()`
- Cosmos handles paging; iterate until HasMoreResults is false.

---

10. How is exception handling and retry logic implemented?

- Catch CosmosException for specific status codes:

```
catch (CosmosException ex) when (ex.StatusCode == HttpStatusCode.TooManyRequests)
{
   await Task.Delay(ex.RetryAfter);
}
```

- SDK includes automatic retry policies; customize via CosmosClientOptions.