Certainly! Here are some other commonly used functions in Parse Cloud code:

1. `.fetch()`: This function is used to retrieve an object from the database based on its objectId. It returns a Promise that resolves to the fetched object. Example:

   ```
   const query = new Parse.Query("MUser");
   const fetchedUser = await query.get(objectId);
   ```

2. `.destroy()`: This function is used to delete an object from the database. It returns a Promise indicating the success or failure of the deletion. Example:

   ```
   const query = new Parse.Query("MUser");
   const userToDelete = await query.get(objectId);
   await userToDelete.destroy();
   ```

3. `.find()`: This function is used to execute a query and retrieve multiple objects that match the query criteria. It returns a Promise that resolves to an array of matching objects. Example:

   ```
   const query = new Parse.Query("MUser");
   query.equalTo("LastName", "Smith");
   const matchingUsers = await query.find();
   ```

4. `.first()`: This function is similar to `.find()`, but it returns only the first matching object. It returns a Promise that resolves to the first matching object or `undefined` if no matches are found. Example:

   ```
   const query = new Parse.Query("MUser");
   query.equalTo("LastName", "Smith");
   const firstUser = await query.first();
   ```

5. `.count()`: This function is used to count the number of objects that match a query. It returns a Promise that resolves to the count. Example:

   ```
   const query = new Parse.Query("MUser");
   query.equalTo("LastName", "Smith");
   const userCount = await query.count();
   ```

These are just a few examples of commonly used functions in Parse Cloud code. The Parse SDK provides a rich set of functions to interact with the Parse Server's database, allowing you to perform various operations like querying, updating, and deleting objects.

Certainly! Here are a few more functions commonly used in Parse Cloud code:

1. `.get()` - This function is used to retrieve a single object from the database based on its objectId. It returns a Promise that resolves to the fetched object. Example:

```
const query = new Parse.Query("MUser");
const fetchedUser = await query.get(objectId);
```

2. `.saveAll()` - This function is used to save multiple objects to the database in a single request. It takes an array of objects and returns a Promise indicating the success or failure of the save operation. Example:

   ```
   const obj1 = new Parse.Object("MyClass");
   obj1.set("key", "value1");

   const obj2 = new Parse.Object("MyClass");
   obj2.set("key", "value2");

   const objectsToSave = [obj1, obj2];
   await Parse.Object.saveAll(objectsToSave);
   ```

3. `.find()` with query constraints - Apart from retrieving all objects, you can also use query constraints to retrieve objects that meet specific conditions. Example:

   ```
   const query = new Parse.Query("MUser");
   query.equalTo("isVerified", true);
   query.limit(10); // Limit the number of results to 10
   const verifiedUsers = await query.find();
   ```

4. `.each()` - This function is used to iterate over each object that matches a query. It takes a callback function as an argument, which is called for each object. Example:

   ```
   const query = new Parse.Query("MUser");
   query.greaterThan("age", 18);
   query.each((user) => {
     console.log(user.get("FirstName"));
   });
   ```

5. `.addUnique()` - This function is used to add unique items to an array field in an object. It ensures that duplicate values are not added to the array. Example:

   ```
   javascriptCopy codeconst obj = new Parse.Object("MyClass");
   obj.addUnique("tags", "tag1");
   obj.addUnique("tags", "tag2");
   await obj.save();
   ```

These additional functions provide further flexibility in querying, manipulating, and managing objects within the Parse Server's database.