CS 327E Project 5, due Thursday, 10/19.

This project makes use of the Shopify public data, the same data we used in Project 3.

The objectives of this project are to redesign the Shopify schema for Firestore, create the database objects (collections and subcollections) according to your new schema design, and populate the documents with the Shopify data.

Using Lucidchart, create an ERD of your Firestore schema. The schema should be modeled according to our design guidelines (see lecture slides) and according to the access patterns given below:

Access patterns:

1. Get apps by category (Category.title)
2. Get apps with highest review_count
3. Get pricing plan details by app (Apps.id)
4. Get key benefits by app (Apps.id)

Note that these access patterns are identical to the ones we used in Practice Problem 1. As there is no access pattern concerning the Reviews entity, you may choose to model reviews either as a top-level collection or as a subcollection of Apps.

Ensure that your diagram captures both the field names and types for each Firestore collection and subcollection. Draw the appropriate relationships between collections. If you are unsure of what type of relationship exists between two collections, consult the Postgres schema you produced for Project 3 and/or sample the data.

For readability, please use one background color to identify the collections in the diagram and a different color to identify the subcollections. Follow the college example from class for other formatting and style guidance.

Export your ERD as a `pdf` file and name it `shopify-firestore-erd.pdf`.

Create a new jupyter notebook and name it `project5.ipynb`. Implement the following logic in your `project5.ipynb` notebook:

- Download the dataset to your notebook instance:
  `gsutil cp gs://cs327e-open-access/fs_shopify.zip .`
- Download Firestore code samples to your notebook instance:
  `gsutil cp gs://cs327e-open-access/fs_samples.zip .`

- Create the Firestore collections and subcollections based on your data model and populate them with the Shopify records.

- Retrieve the number of documents in each collection and subcollection using `count()`. See code sample `read_count.py`. The desired output should be a single number for each collection and subcollection. Hint: you want to aggregate all the individual subcollection counts for each unique subcollection name.

- List the top 10 "Productivity" apps (whose `categories.title` = "Productivity") sorted by their rating in descending order. Return the `id`, `title`, `developer`, `rating` and `reviews_count` for those apps. Limit the results to the first 10 records. Note: this query refers to access pattern #1.

- List the 10 apps with the highest number of reviews (based on `apps.review_count`). Return the `id`, `title`, `developer`, `rating` and `reviews_count` for those apps. Order the results by `reviews_count` in descending order.
  Note: this is query refers to access pattern #2.

CS 327E Project 5 Rubric
Due Date: 10/19/23

Create an ERD for the shopify data in Firestore. Include field names, data types, and ids for the collections and subcollections. Draw the proper relationships between collections and subcollections. **-3** for each missing field name, data type or id **-3** for each missing or incorrect relationship between entity types	20
Create the Firestore database objects that are represented in your ERD: **-5** for each collection which does not match its entity specification **-4** for each subcollection which does not match its entity specification **-3** for each field which does not match its entity specification **-2** for each id which does not match its entity specification	20
Populate each collection with the appropriate shopify records and retrieve a count of the number of documents per collection and subcollection. **-5** for each empty collection or subcollection **-3** for each collection which has missing documents **-3** for each missing or incorrect count **-2** for each count not using the `count()` function	40
List the top 10 productivity apps with the highest rating. **-3** incorrect or missing filter **-3** incorrect or missing order by **-3** incorrect number of results returned **-3** incorrect or missing fields in results	10
List the 10 apps with the highest number of reviews. **-3** incorrect or missing order by **-3** incorrect or missing fields in results **-3** incorrect number of results returned	10
`shopify-firestore-erd.pdf` pushed to your group's private repo on GitHub. Your project **will not** be graded without this submission.	**Required**
`project5.ipynb` pushed to your group's private repo on GitHub. Your project **will not** be graded without this submission.	**Required**

`submission.json` submitted into Canvas. Your project **will not** be graded without this submission. The file should have the following schema:
 	
`{`
`  "commit-id": "your most recent commit ID from GitHub",`
`  "project-id": "your project ID from GCP"`
`}`
 	
Example:
	Required

<pre>{
    "commit-id": "dab96492ac7d906368ac9c7a17cb0dbd670923d9",
    "project-id": "some-project-id"
}</pre>	

Total Credit:	**100**

FAQs

I’m seeing a lower count for my subcollection/collection than what’s present in the CSV.

Solution: The count might mismatch the expected number if your documents don’t have a unique identifier attached to them. You can ensure a unique identifier in the following two ways:

1. Concatenating multiple fields to ensure uniqueness

or

2. Using the UUID library to generate a unique id. For example, if you were dealing with the key benefits sub-collection:

```
import uuid

benefits_id = str(uuid.uuid4())
benefits_ref = apps_ref.collection('key_benefits').document(benefits_id)
```