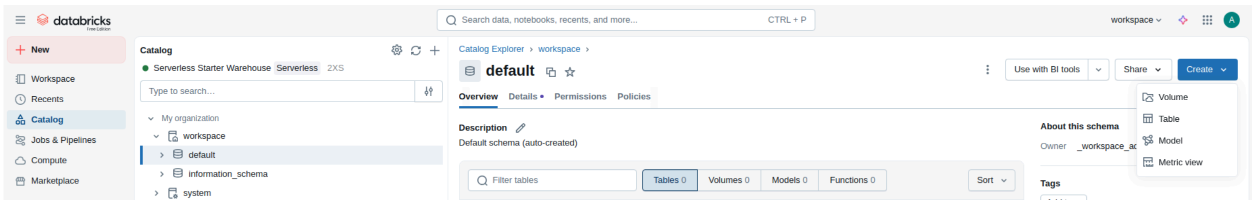
Homework Assignment #5

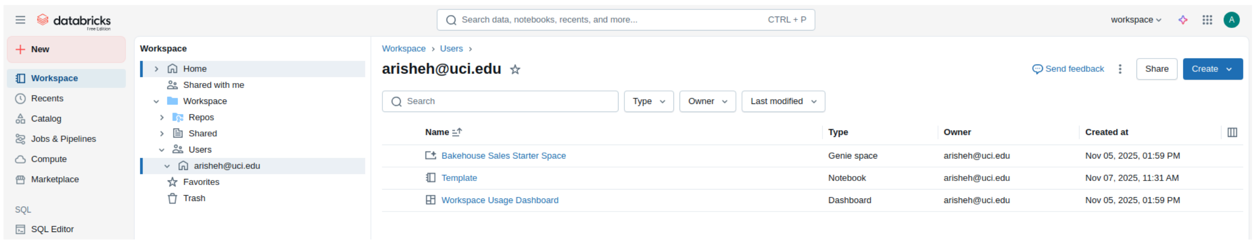
Deadline: 11/20/2026 (Thursday) 11:45PM

Setup

1. Go to databricks.com and create a new account and select the free edition. Databricks is the company that hosts Apache Spark as a cloud service.
2. Once you select the free edition, it automatically creates a serverless instance to explore Spark features.
3. Download the sample data with two JSONL files. It is for testing purposes.
4. You need to create one table per JSONL file. First from the left menu: go to Catalog and select workspace. Then choose the default schema: on the top right of the schema, click the Create option and select Table.



5. Upload the users.jsonl file and wait for the application to process the data. The system should automatically identify the structure of the data.
6. Finally, check that the table name is "users" and then create the table. Repeat these steps for the "purchases.jsonl" file.
7. Now you have created the tables. Next is to explore features provided by Spark. Download template.ipynb and go to workspace -> Workspace -> Users -> <Your email> and upload the template file there.



Name	Type	Owner	Created at
Bakehouse Sales Starter Space	Genie space	arisheh@uci.edu	Nov 05, 2025, 01:59 PM
Template	Notebook	arisheh@uci.edu	Nov 07, 2025, 11:31 AM
Workspace Usage Dashboard	Dashboard	arisheh@uci.edu	Nov 05, 2025, 01:59 PM

8. Open the file and run the cells to explore and see some of the features of the Spark engine.

Let's go…

You have been working all quarter long on different database systems and experiencing their varying environments. As the quarter progresses, ZotPets's grooming and pet supply business has taken off, with a huge growth in their user base and sales.

Given the explosion in the data size, you have decided to explore Apache Spark, a powerful framework that enables you to execute parallel query tasks to expedite big data processing.

Get Ready... (and set)

To start, follow the same process you did for "users" to import the full data and create one table per JSONL file. You need to delete the sample ones. Download the full dataset and upload them into your Databricks catalog.

It's time to explore the new system! Begin your Big Data analytics adventure by running the provided setup cell and then exploring the schema of the preloaded DataFrames and views.

1. Start by running some schema description statements to better understand the preloaded data.

A. First, explore the schema for the `users_df` DataFrame. Write a Python snippet using Spark DataFrames to show the schema.

B. What is the data type of the `pets` field in the `users_df` DataFrame?

C. Now let's see how to view a schema using SQL in Spark. Write a SQL statement to view the schema of the `zotpets_products` table.

D. What is the data type of the `price` field in the `zotpets_products` table?

Hint: in the Databricks Notebook file, you can use SQL queries by adding `%sql` to the top of the cell. It gives you a better interface. For example:

```sql
%sql
DESCRIBE TABLE products
```

2. After scratching the surface of Spark, you are eager to start writing queries and analyzing the data!

For the following questions, you must provide **TWO** ways to get the answer:

- First, by providing the **DataFrame API code** (Python).
- Second, by providing the equivalent **SQL statement** (using `%sql`).

A. To start your Spark journey, you would like to view all the details of a specific user. Find and display all information for the user whose `user_id` is `1`.

B. Find the total number of "Shampoo" products sold by each groomer. Include the groomer's `user_id` and the total count of shampoo products sold. Sort the results by the total count in descending order, and limit the results to the top 5 groomers.

C. Find the top 3 busiest days of December 2024. That is, determine the total number of appointments scheduled for each day in December 2024. Print the date and the total appointment count. Sort the results in descending order of the appointment count.

D. You have heard that Spark can deal with array data well, and you want to explore this feature. Your manager wants to know what kinds of pets your customers have. Find the most popular pet `species` across all pet owners. Print the `species` and the total count of pets for that species. Sort the results in descending order by the count, and show the top 5.

E. `Join` is one of the most interesting operations for data analysis. You want to find out which pet owners are booking appointments. Find and print the `appointment_id`, the `appointment_datetime`, and the `email` of the pet owner for all appointments scheduled on or after November 1st, 2025 (inclusive). Sort the results in ascending order of the `appointment_datetime`, and limit the results to the first 5.

F. Now let's try a query that requires joining three tables. Your manager wants to identify the top-performing groomers in terms of product sales. Find the top 3 groomers who have generated the most revenue from selling products. Print the groomer's `name`, `email`, and their `total_revenue`. Sort the results in descending order of `total_revenue`.

G. You've mastered the Spark basics and are ready for a complex "boss-level" query. Your manager wants to know if appointments drive product sales, and if this differs by the type of pet.

First, find all pet owners who had at least one appointment in 2024. Then, for only those owners, calculate the total product revenue (from all their purchases, regardless of date) and break this total down by the `species` of the pet that had the 2024 appointment. This will show, for example, "Dog" owners who had a 2024 appointment generated $X in total product sales.

Print the `species`, the total `revenue`, and the `unique_owner_count` for each species. Sort by `revenue` in descending order. *Hint: This query requires joining all four tables and using `explode()`.*

3. Discussion Question - No Code Required

The final query (2.G) was a heavy, 4-table analytical join that also required "exploding" semi-structured array data. This type of query is Spark's specialty.

In a Markdown cell, briefly explain (in a few sentences for each) why this *specific query* would be difficult or slow to run on the following systems, assuming a 500GB dataset:

- **A) PostgreSQL (Relational DB):** How would storing the `pets` array (semi-structured data) make this query more complex?
- **B) MongoDB (Document DB):** How would you get data from four different "collections" (our tables)? What is the major weakness of document databases that this query exposes?
- **C) Cassandra (Key-Value/Wide-Column DB):** Cassandra is built for massive scale. Why would it be a *terrible* choice for this kind of ad-hoc analytical query?

What to submit?

Please create a structured Notebook with markdown cells for explanation and clean coding cells. Finally export the Notebook as PDF and upload.