CS 327E Final Project: Milestone 1, due Thursday 11/30

This milestone has 4 parts. Each part asks you to generate some sample test data and populate a different database that we previously studied in this course. The next milestone will build upon this work by joining tables that span multiple databases via Trino.

1. In JupyterLab, install a Python database connector for MySQL, Postgres, MongoDB, and BigQuery:

   `pip install mysql-connector-python` (documentation)
   `pip install psycopg[binary]` (documentation)
   `pip install pymongo[srv]`(documentation)
   `pip install google-cloud-bigquery` (documentation)

Part 1: MySQL

2. Create a Jupyter notebook and name it `final-project-mysql.ipynb`. Implement the following logic in your notebook using a combination of SQL and Python code. To help you get started, please review the code samples.

3. Create a database in MySQL with a table by the name of **shopper**. Define the shopper table based on this description:

Constraint	Column Name	Description
Primary Key	cust_id	Customer Identifier
	first_name	First Name
	last_name	Last Name
	company	Company Name
	street_1	Street Address 1
	street_2	Street Address 2 (appt number, suite number, box number)
	city	City Name
	county	County Name
	state	State or Province
	zip	Zip Code or Postal Code
	phone_1	Phone Number 1

	phone_2	Phone Number 2
	email	Email Address

Note: you'll need to assign an appropriate data type to each field in the table.

4. Write a data generator that populates the shopper table with 50 unique records. Your code should produce the following output:

```
50 records written into shopper table
```

Part 2: Postgres

5. Create a Jupyter notebook and name it `final-project-postgres.ipynb`. Implement the following logic in your notebook using a combination of SQL and Python code. To help you get started, please review the code samples.

6. Create a schema in Postgres with a table by the name of **reservations**. Define the reservations table based on this description:

Constraint	**Column**	**Description**
Primary Key	res_id	Reservation Identifier
Join Key	cust_id	Customer Identifier
	prp_nm	Property Name
	prp_ch	Property Chain
	adr_line_1	Street Address 1
	adr_line_2	Street Address 2 (appt number, suite number, box number)
	city	City Name
	state	State or Province Name
	postal_cd	Zip Code or Postal Code
	lat	Latitude
	long	Longitude

Join Key	cnt_code	Country Code
	arr_date	Arrival Date
	dep_date	Departure Date
	pmt_amt	Payment Amount

Note: you'll need to assign the appropriate data type to each field in the table.

7. Write a data generator that populates the **reservations** table with 100 unique records. Your code should produce the following output:

```
100 records written into reservations table
```

Part 3: MongoDB

8. Create a Jupyter notebook and name it `final-project-mongodb.ipynb`. Implement the following logic in your notebook using a combination of SQL and Python code. To help you get started, please review the code samples. Remember to grant your JupyterLab VM access to your shared MongoDB cluster on Atlas.

9. Create a database in MongoDB with a collection by the name of **ticketing**. Populate the ticketing collection with 100 unique records that conform to this schema:

Constraint	**Field**	**Description**
Primary Key	tck_id	Ticket Identifier
Join Key	cust_id	Customer Identifier
	airline	Airline Name
	flight_nm	Flight Number
	dep_airport	Departure Airport
	arr_airport	Arrival Airport
	dep_date	Departure Date
	dep_time	Departure Time
	arr_date	Arrival Date

	arr_time	Arrival Time
	stops	Number of stops
	tik_amt	Ticket Amount / Price
Join Key	curr_code	Currency Code

Note: you'll need to assign appropriate types to each field in the collection.

Your code should produce the following output:

```
100 documents written into ticketing collection
```

Part 4: BigQuery

10. Create a Jupyter notebook and name it `final-project-bigquery.ipynb`. Implement the following logic in your notebook using a combination of SQL and Python code. To help you get started, please review the code samples.

11. Create a dataset in BigQuery with a table by the name of **currency**. Define the currency table based on this description:

Constraint	**Column**	**Description**
Primary Key and Join Key	curr_code	Currency Code
	curr_name	Currency Name
Join Key	cntry_code	Country Code
	cntry_name	Country Name

Note: you'll need to assign an appropriate data type to each field in the table.

12. Write a data generator that populates the **currency** table with 30 unique records.

Your code should produce the following output:

```
30 records written into currency table
```

Additional Notes

- Customer identifiers need to match up between shopper, reservations, and ticketing tables so that they can be joined.
- Country codes need to match up between the reservations and currency tables so that they can be joined.
- Currency codes need to match up between the ticketing and currency tables so that they can be joined.
- Address fields can be inconsistent within the same record. For example, a county can be unrelated to the city of the same address.
- Use the Faker library to generate person names, addresses, currencies, etc.
- Use faker_airtravel to generate airports and stops.
- Use `randrange()` and `timedelta()` to generate date ranges.

CS 327E Final Project Milestone 1 Rubric
Due Date: 11/30/23

Code block that creates the **shopper** table in a MySQL database. **-1** incorrect table name **-1** for each missing column or incorrect column name **-1** for each field type mismatch (e.g. assigned VARCHAR instead of CHAR, etc.) **-1** incorrect primary key specification (e.g. sequence generator missing or incorrect) **-1** missing or incorrect table drop if exists	10
Code block that generates the **shopper** records and writes them into previously created table in MySQL. **-3** code has syntax errors and/or doesn't run **-3** Faker library not used to generate names, addresses, phone numbers, etc. **-1** for each value missing from insert statement **-1** incorrect output produced **-1** missing try/except blocks	15
Code block that creates the **reservations** table in a Postgres database. **-1** incorrect table name **-1** for each missing column or incorrect column name **-1** for each field type mismatch (e.g. assigned VARCHAR instead of CHAR, etc.) **-1** incorrect primary key specification (e.g. sequence generator missing or incorrect) **-1** missing or incorrect table drop if exists	10
Code block that generates the **reservations** records and writes them into previously created table in Postgres. **-3** code has syntax errors and/or doesn't run **-3** Faker library not used to generate names, addresses, phone numbers, etc. **-1** for each value missing from insert statement **-1** incorrect output produced **-1** missing try/except blocks	15
Code block that generates the **ticketing** documents and writes them into the **ticketing** collection in a MongoDB database. **-3** code has syntax errors and/or doesn't run **-1** incorrect collection name **-1** for each missing field or incorrect field name **-1** for each field type mismatch (e.g. used a String instead of Date, etc.) **-3** Faker library not used to generate airports, stops **-1** for each value missing from insert statement **-1** incorrect output produced **-1** missing try/except blocks	25
Code block that creates the **currency** table in a BigQuery dataset. **-1** incorrect table name **-1** for each missing column or incorrect column name **-1** missing or incorrect table replace statement	7

Code block that generates the **currency** records and writes them into previously created BigQuery table. **-3** code has syntax errors and/or doesn't run **-3** Faker library not used to generate currencies and/or countries. **-1** for each value missing from insert statement **-1** incorrect output produced **-1** missing try/except blocks	10
Referential integrity between the previously created tables across MySQL, Postgres, MongoDB, and BigQuery. **-2** values in postgres.reservations.cust_id don't exist in mysql.shopper.cust_id **-2** values in postgres.reservations.cnt_code don't exist in bigquery.currency.cnt_code **-2** values in mongodb.ticketing.cust_id don't exist in mysql.shopper.cust_id **-2** values in mongodb.ticketing.curr_code don't exist in bigquery.currency.curr_code	8
`final-project-mysql.ipynb`, `final-project-postgres.ipynb`, `final-project-mongodb.ipynb`, and `final-project-bigquery.ipynb` pushed to your group's private repo on GitHub. Your milestone **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:
 	
`{`
`  "commit-id": "dab96492ac7d906368ac9c7a17cb0dbd670923d9",`
`  "project-id": "some-project-id"`
`}`
	Required

Total Credit:	**100**