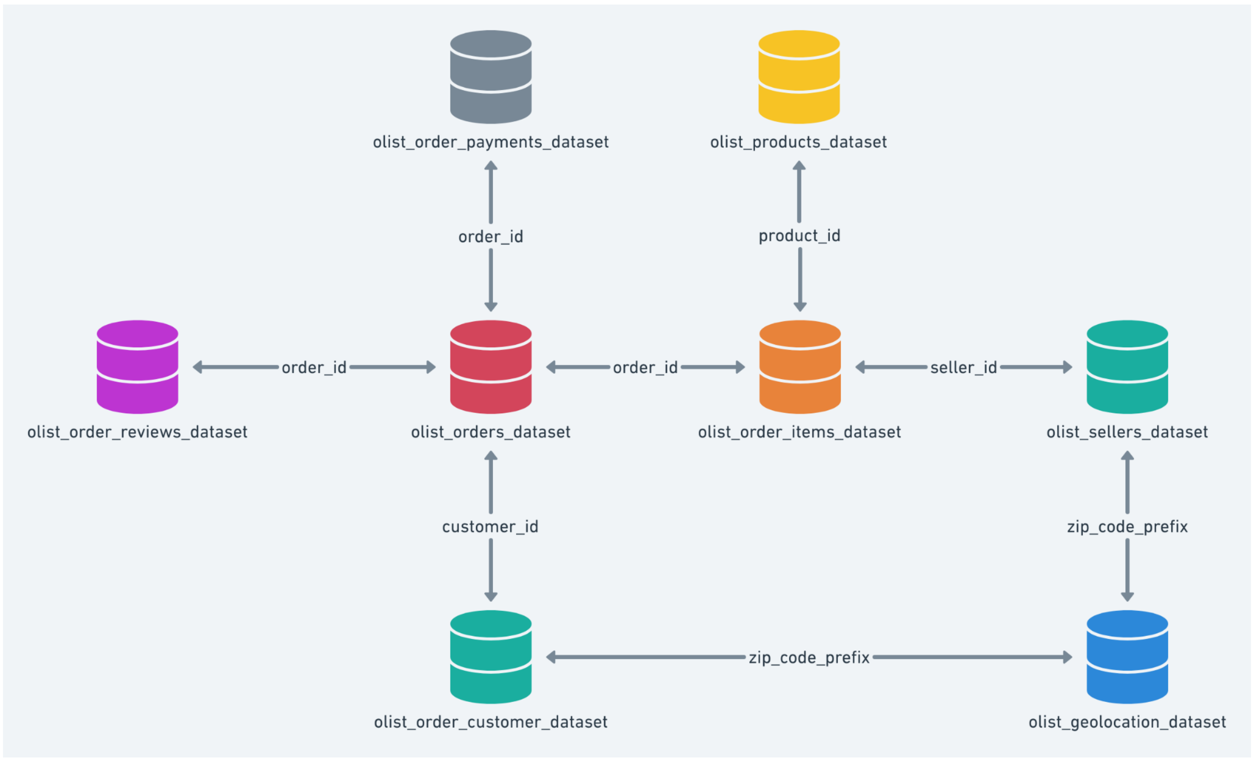
Part 1

Let's say that one of our clients is preparing to migrate to a new order management system. While there's no live order data from the new system yet, we want to get things set up so that the client can have a smooth transition.

Conveniently, our data integration provider has a sample data set that we can use to start getting the data warehouse set up. After speaking with the product manager and analyst who are also working on this project, you know that the data model that summarizes orders by day is the highest priority.

You get a copy of the data schema, and you're ready to get started:



Accessing the Data

To access the datasets, you can use any email address to sign up at https://modeanalytics.com/signup.

The tables you'll need can all be found under **brooklyndata.olist_**, and you'll find them hosted on our public datasets page in Mode: https://app.mode.com/brooklyndata/tables

More about Mode:

- Start a query by clicking the large plus sign in the upper right hand corner of the main page. This database uses PostgreSQL.
- More on the Mode query editor: https://mode.com/help/articles/reports/#querying-data

Exercise

1. Write a single query that produces the following metrics at order purchase date grain in descending date order:
    a. Total orders
    b. Total customers making orders
    c. Total revenue
    d. Average revenue per order
    e. Top 3 product categories by revenue (by day)
    f. Percent of day's revenue associated with each of the top 3 product categories

    The columns should be named and formatted as follows:

Column Name	Format/Type
order_purchase_date	date
orders_count	number
customers_making_orders_count	number
revenue_usd	number - two decimal places
average_revenue_per_order_usd	number - two decimal places
top_3_product_categories_by_revenue	string - comma separated list, in descending order of revenue
top_3_product_categories_revenue_percentage	string - comma separated list of numbers to two decimal places, in descending order of revenue percentage

2. Using code comments, explain how your query produces each metric.

3. Now that you've finished making a pass at your query, it's always a good idea to get a second set of eyes on your code. **Write some notes to your peer reviewer** that give some context on the dataset, identify anything notable about your approach, and specify any open questions or concerns you might have about your approach.
4. Sometimes a client's data is not straightforward, and you may need to make some assumptions and validate them later with the client. If that was the case, **write some notes for your client**, explaining your assumptions, and asking any clarifying questions you may have.

Note: you may translate the product categories if you'd like (with **brooklyndata.product_category_name_translation**), but that's not necessary or expected for this exercise.

We use a SQL style guide at Brooklyn Data for consistency and ease of review, please take a read on GitHub.

Part 2

Let’s say that one of our clients needs to load data from a number of CSV files into a particular table in a Snowflake database. The CSV files are over 100GB each. The first row of each CSV file contains column headers which match the names of the columns in the database table, though the columns in the CSV files are not necessarily in the same order as in the table.

Exercise

1. **Write a function** in Python which takes as arguments a Snowflake connection, the fully qualified name of the table, and a list of the CSV file paths, and loads the data from those CSV files into the table in the Snowflake database. The required signature is as follows:

```
def load_csvs_to_snowflake_table(
   conn: snowflake.connector.connection.SnowflakeConnection,
   fully_qualified_table_name: str,
   csv_file_paths: List[str],
):
```

2. It’s always a good idea to get a second set of eyes on your code, so imagine you were going to submit this function for peer review. **Write some comments to your peer reviewer** that give some context on what you were trying to achieve, identify anything notable about your approach, and specify any open questions or concerns you might have.
3. The client would like to be able to use the function to upload CSVs from a folder on their computer. **Write a script in a second file** which calls this function, with placeholders where they can add their Snowflake credentials and the path to a folder on their computer containing the CSV files. Provide some instructions on how to use the script, including any prerequisite steps such as installing libraries or packages. Assume the client has Python 3.8 installed already.
4. In words, describe any changes you’d make to this function for use in production.