Data Engineering, ETL and Integrations

Introduction

This exercise is broken into three parts. The first two will involve SQL and data modeling. The third will focus on object oriented programming in Python.

Part One

We'll begin by defining two tables, `items` and `orders`. The column names are, for the most part, self explanatory. All of the items in a customer's order will have the same `order_id`, and each customer can have multiple orders. The tables can be joined on `items.id` and `orders.item_id`.

```
CREATE TABLE items
(
  id      SERIAL NOT NULL
    CONSTRAINT items_pkey
    PRIMARY KEY,
  upc     INTEGER,
  name    TEXT,
  size    TEXT,
  price   DOUBLE PRECISION,
  taxable BOOLEAN,
  sold_by TEXT
);
CREATE UNIQUE INDEX items_id_uindex
  ON items (id);
```

```
INSERT INTO items (id, upc, name, size, price, taxable, sold_by)
  VALUES (4, 30273, 'Apple', 'per lb', 0.99, false, 'weight');
INSERT INTO items (id, upc, name, size, price, taxable, sold_by)
  VALUES (1, 4738561, 'Milk', '1 gallon', 2.89, false, 'count');
INSERT INTO items (id, upc, name, size, price, taxable, sold_by)
  VALUES (2, 8897585, 'Bread', '1 loaf', 3.5, false, 'count');
INSERT INTO items (id, upc, name, size, price, taxable, sold_by)
  VALUES (5, 3342, 'Banana', '1 each', 0.69, false, 'count');
INSERT INTO items (id, upc, name, size, price, taxable, sold_by)
  VALUES (6, 908345, 'Cashews', '16 oz', 6.99, false, 'count');
INSERT INTO items (id, upc, name, size, price, taxable, sold_by)
  VALUES (3, 908347, 'Yogurt', '1 container', 1.25, true, 'count');
INSERT INTO items (id, upc, name, size, price, taxable, sold_by)
  VALUES (7, 30273, 'Apple', 'per lb', 1.09, false, 'weight');
```

```
INSERT INTO items (id, upc, name, size, price, taxable, sold_by)
  VALUES (8, 3342, 'Banana', 'per lb', 0.56, true, 'weight');



CREATE TABLE orders
(
  id           SERIAL NOT NULL
    CONSTRAINT orders_pkey
    PRIMARY KEY,
  order_id INTEGER,
  customer_id  INTEGER,
  item_id      INTEGER
    CONSTRAINT orders_items_id_fk
    REFERENCES items,
  name         TEXT,
  phone        TEXT,
  address      TEXT,
  delivered    BOOLEAN,
  quantity     DOUBLE PRECISION
);
CREATE UNIQUE INDEX orders_id_uindex
  ON orders (id);



INSERT INTO orders (id, order_id, customer_id, item_id, name,
  phone, address, delivered, quantity)
  VALUES (3, 23, 3456, 1, 'Bob', null, null, false, 2);
INSERT INTO orders (id, order_id, customer_id, item_id, name,
  phone, address, delivered, quantity)
  VALUES (4, 23, 3456, 2, 'Bob', null, null, false, 1);
INSERT INTO orders (id, order_id, customer_id, item_id, name,
  phone, address, delivered, quantity)
  VALUES (5, 23, 3456, 3, 'Bob', null, null, false, 6);
INSERT INTO orders (id, order_id, customer_id, item_id, name,
  phone, address, delivered, quantity)
  VALUES (6, 23, 3456, 5, 'Bob', null, null, false, 3);
INSERT INTO orders (id, order_id, customer_id, item_id, name,
  phone, address, delivered, quantity)
  VALUES (7, 89, 2239, 4, 'Alice', null, null, false, 2);
INSERT INTO orders (id, order_id, customer_id, item_id, name,
  phone, address, delivered, quantity)
  VALUES (8, 89, 2239, 6, 'Alice', null, null, false, 1);
INSERT INTO orders (id, order_id, customer_id, item_id, name,
  phone, address, delivered, quantity)
  VALUES (9, 65, 2239, 1, 'Alice', null, null, true, 1);
INSERT INTO orders (id, order_id, customer_id, item_id, name,
  phone, address, delivered, quantity)
  VALUES (10, 65, 2239, 3, 'Alice', null, null, true, 4);
INSERT INTO orders (id, order_id, customer_id, item_id, name,
```

```
    phone, address, delivered, quantity)
    VALUES (11, 65, 2239, 2, 'Alice', null, null, true, 1);
```

Questions

1. How would you find the order total (defined above) and the average order total per customer?
2. There are duplicates in the `items` table. Before deleting the duplicate records from the table, we need to update the `orders.item_id` field to ensure that there are no references to the records we're going to delete. Write a query to update the `orders` table to swap the `item_id` values if necessary. What do you think is reasonable business logic to determine which duplicate record to keep?
3. Suppose you were redesigning the model for these tables. Let's say that we were expanding and now need to support variations in pricing and availability across locations. For example:
    - `item 1` at `store 1` is $2.99
    - `item 1` at `store 2` is $2.50

    If we were to simply add a `store_location` column to the `items` table, we would end up duplicating a lot of data. This could cause issues when scaling. Similarly, the `orders` table has a lot of duplicated data.

We're looking more for discussion and diagrams for this question and not necessarily SQL.

Part Two

Properly categorizing products can be a complex task. Below we have two tables. `imported_products` is data from Shipt's newest partner. `category_map` is a mapping table created by the Catalog team to organize products. The tables can be joined on `imported_products.category` and `category_map.id`.

```
-- Postgres ver 9.6.1
CREATE TABLE imported_products
(
  product_id      INTEGER NOT NULL
    CONSTRAINT products_pkey
    PRIMARY KEY,
  upc     INTEGER,
  name    TEXT,
  size    TEXT,
  category    TEXT
);
```

```
INSERT INTO imported_products (product_id, upc, name, size, category)
  VALUES (4, 30273, 'Apple', 'per lb', '2_PRODUCE_FRUITS');
INSERT INTO imported_products (product_id, upc, name, size, category)
  VALUES (1, 4738561, '2% Milk', '1 gallon', '54_DAIRY_MILK');
INSERT INTO imported_products (product_id, upc, name, size, category)
  VALUES (2, 8897585, 'Skim Milk', '1 gallon', '55_DAIRY_LOWFATMILK');
INSERT INTO imported_products (product_id, upc, name, size, category)
  VALUES (5, 3342, 'Banana', '1 each', '2_PRODUCE_FRUITS');
INSERT INTO imported_products (product_id, upc, name, size, category)
  VALUES (6, 908345, 'Organic Peas', '1 head', '4_PRODUCE_ORG_VEGETABLES');
INSERT INTO imported_products (product_id, upc, name, size, category)
  VALUES (3, 908347, 'Yogurt', '1 container', '52_DAIRY_YOGURT');
INSERT INTO imported_products (product_id, upc, name, size, category)
  VALUES (7, 30273, 'Oranges', 'per lb', '3_PRODUCE_VEGETABLES');
INSERT INTO imported_products (product_id, upc, name, size, category)
  VALUES (8, 3342, 'Potatoes', 'per lb', '3_PRODUCE_VEGETABLES');
INSERT INTO imported_products (product_id, upc, name, size, category)
  VALUES (9, 30275, 'Organic Apple', 'per lb', '2_PRODUCE_ORG_FRUITS');
```

```
-- Postgres ver 9.6.1
CREATE TABLE category_map
(
  id                    INTEGER NOT NULL
    CONSTRAINT category_pkey
    PRIMARY KEY,
  category_key          TEXT,
```

```
  category_name         TEXT,
  parent_category_id    INTEGER
);
```

```
INSERT INTO category_map (id, category_key, category_name, parent_category_id)
  VALUES (600, '1_PRODUCE', 'Produce', null);
INSERT INTO category_map (id, category_key, category_name, parent_category_id)
  VALUES (604, '2_PRODUCE_FRUITS', 'Produce/Fruits', 600);
INSERT INTO category_map (id, category_key, category_name, parent_category_id)
  VALUES (608, '3_PRODUCE_VEGETABLES', 'Produce/Vegetables', 600);
INSERT INTO category_map (id, category_key, category_name, parent_category_id)
  VALUES (606, '4_PRODUCE_ORG_VEGETABLES', 'Produce/Vegetables', 600);
INSERT INTO category_map (id, category_key, category_name, parent_category_id)
  VALUES (800, '50_DAIRY', 'Dairy', null);
INSERT INTO category_map (id, category_key, category_name, parent_category_id)
  VALUES (803, '52_DAIRY_YOGURT', 'Dairy/Yogurt', 800);
INSERT INTO category_map (id, category_key, category_name, parent_category_id)
  VALUES (801, '54_DAIRY_MILK', 'Dairy/Milk', 800);
INSERT INTO category_map (id, category_key, category_name, parent_category_id)
  VALUES (802, '55_DAIRY_LOWFATMILK', 'Dairy/Milk', 800);
```

```
-- Postgres ver 9.6.1
CREATE TABLE product_categorizations
(
  id                    INTEGER NOT NULL
    CONSTRAINT category_pkey
    PRIMARY KEY,
  product_id            INTEGER NOT NULL,
  category_id           INTEGER NOT NULL
);
```

Questions

1. On every import, the data pipeline inserts the parent and child categories to a `product_categorizations` table for each `product_id`. A Shipt Catalog Analyst discovers that the Oranges are being listed with Vegetables instead of Fruits and added the correct mapping to the `product_categorizations` table. The new Partner says this is a common issue and it may take several weeks to correct the data feed. How would you address this issue? What modifications would you make? You may edit any functions and/or create & modify tables.

2. Shipt Engineering just launched a new feature to support taxonomies that are now 3 tiers instead of a 2 tier parent-child. Some product mappings will stay two levels, but others will now get grandchild categories (example: Vegetables can be "Regular" and "Organic"). How would you structure the data model or imports to support this? What would a query to look like to find mappings for a product?

3. One of the products ("Organic Apples") doesn't have a category mapping. Without a mapping, the item won't appear in the Shipt catalog. How would you solve this problem, knowing that new product categories regularly appear in the pipeline without warning?

We're looking more for discussion and diagrams for these questions and not necessarily blocks of SQL.

Part Three

The preferred language here is Python, but if you are comfortable in another language that is also ok. Please provide clear instructions on how to run your code or any dependencies if needed. Please validate your code before you send it to us.

Questions

Using the Star Wars API located here: https://swapi.co/

1. Retrieve a list of all people.
2. In parallel, build a list of names of all planets and ships associated with each person.
3. What species live on multiple planets?

Expected output: Running the code should get me the list of all people and names of all planets and ships associated with each person.