Data Assignment - Reunion

Jagannath V V

Problem 1: Data Modelling (related to Problem 2)

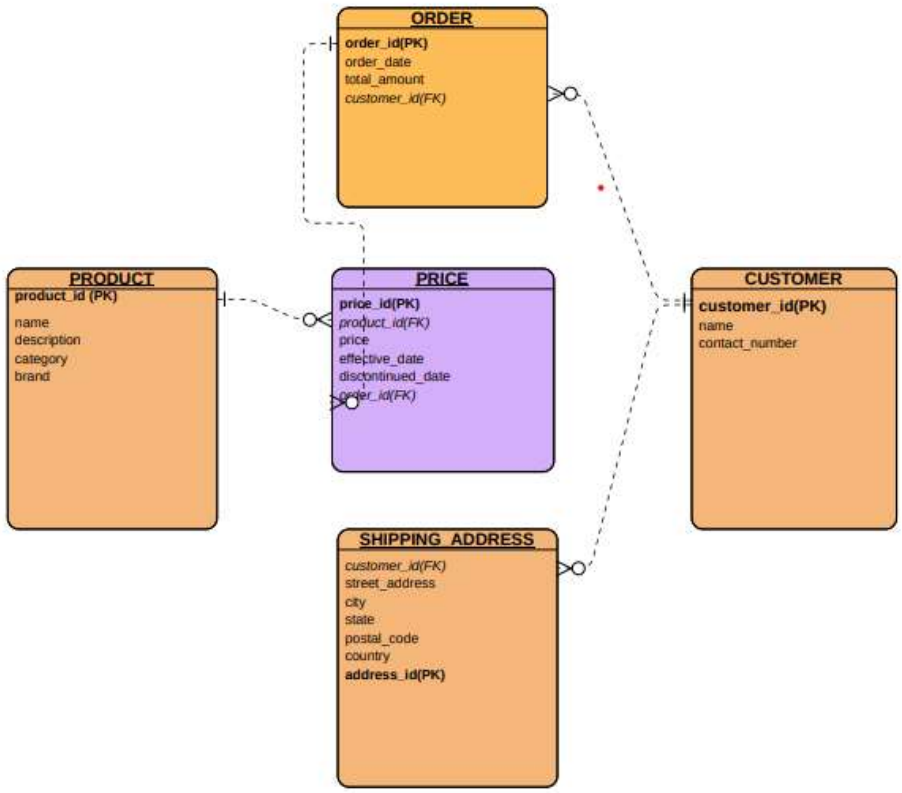
Imagine you are designing a database for an e-commerce platform. The database should store information about products, customers, orders, etc. (this is only indicative; please feel free to create tables as per your imagination). Customer details such as shipping address, contact number etc. can change over time. Each product can have multiple variants based on its attributes. Each variant may be launched at various times, then discontinued and then perhaps relaunched later as per the business requirements. Price of the product and its variants may keep on changing with time. We want to retain the historical information for these changes as well in our schema. You are required to:

```
==========================================================================
```

(Q1). Design a star-schema / snowflake schema model for the above requirements

```
a). Use an entity-relationship diagram (ERD) that represents the
relationships between these entities

b). Include the necessary attributes and primary/foreign key
relationships. Briefly explain your design choices.
```



I used Visual Paradigm Online to create this ER-Diagram

The Entites are :

Products Table:

The "products" table holds essential information about each item available in the e-commerce platform, such as its name, description, and category. It acts as a comprehensive catalog of all products in the system.

Variants Table:

The "variants" table contains specific details of various versions or variations of a product. For instance, if a product comes in different sizes, colors, or other attributes, each variant is stored in this table. The connection to the "products" table is established through a unique product ID, allowing us to associate each variant with its parent product.

Customers Table:

The "customers" table stores key information about each customer, including their name and email address. It serves as a centralized database of all registered users on the e-commerce platform.

Addresses Table:

The "addresses" table contains shipping addresses provided by customers. Each address is linked to a specific customer using a customer ID. Moreover, a flag indicates whether an address is the customer's current shipping address.

Orders Table:

The "orders" table holds important details about each customer order, such as the order date and the total amount spent. Each order is connected to the respective customer through a customer ID.

Order Items Table:

The "order_items" table stores specific information about individual items within an order, including the quantity and price. It allows us to break down an order into its constituent products. Each order item is linked to the corresponding order and the specific product variant purchased, enabling us to track the products included in each order.

==========

(Q2). Generate and insert sample data in the above model. Include the process and code of generating random data in your submission. You data should have:

a)At least 2 years of order history

b)At least 10 products; at least 2 products with variants.

c)At least 10 customers

In [ ]:
```
pip install Fake
```

```
Collecting Fake
  Downloading fake-0.8.tar.gz (4.8 kB)
Collecting fabric>=1.12
  Downloading fabric-3.1.0-py3-none-any.whl (57 kB)
Collecting invoke>=2.0
  Downloading invoke-2.2.0-py3-none-any.whl (160 kB)
Requirement already satisfied: paramiko>=2.4 in c:\users\jagan\anaconda3\lib\site-packag
es (from fabric>=1.12->Fake) (2.7.2)
Requirement already satisfied: decorator>=5 in c:\users\jagan\anaconda3\lib\site-package
s (from fabric>=1.12->Fake) (5.1.0)
Requirement already satisfied: bcrypt>=3.1.3 in c:\users\jagan\anaconda3\lib\site-packag
es (from paramiko>=2.4->fabric>=1.12->Fake) (3.2.0)
Requirement already satisfied: cryptography>=2.5 in c:\users\jagan\anaconda3\lib\site-pa
ckages (from paramiko>=2.4->fabric>=1.12->Fake) (3.4.8)
Requirement already satisfied: pynacl>=1.0.1 in c:\users\jagan\anaconda3\lib\site-packag
es (from paramiko>=2.4->fabric>=1.12->Fake) (1.4.0)
Requirement already satisfied: six>=1.4.1 in c:\users\jagan\anaconda3\lib\site-packages
(from bcrypt>=3.1.3->paramiko>=2.4->fabric>=1.12->Fake) (1.16.0)
Requirement already satisfied: cffi>=1.1 in c:\users\jagan\anaconda3\lib\site-packages
(from bcrypt>=3.1.3->paramiko>=2.4->fabric>=1.12->Fake) (1.14.6)
Requirement already satisfied: pycparser in c:\users\jagan\anaconda3\lib\site-packages
(from cffi>=1.1->bcrypt>=3.1.3->paramiko>=2.4->fabric>=1.12->Fake) (2.20)
Building wheels for collected packages: Fake
  Building wheel for Fake (setup.py): started
  Building wheel for Fake (setup.py): finished with status 'done'
  Created wheel for Fake: filename=fake-0.8-py3-none-any.whl size=7530 sha256=8efadb59d8
10daa03060ff21f78a261a661819b40f8d380ceb873913d2e2dc86
  Stored in directory: c:\users\jagan\appdata\local\pip\cache\wheels\8b\c4\2f\ccaa0d9cc5
2234323bebc47a7d7c31042c79a5b82ced51da3d
Successfully built Fake
Installing collected packages: invoke, fabric, Fake
```

```
Successfully installed Fake-0.8 fabric-3.1.0 invoke-2.2.0
Note: you may need to restart the kernel to use updated packages.
```

Importing Libraries

In [ ]:

```
from faker import Faker
import random
import datetime
```

Solution

In [ ]:

```
fake = Faker()

# Generating sample data for products and variants
products = []
variants = []

for i in range(1, 11):
    product = {
        "id": i,
        "name": fake.word(),
        "description": fake.sentence(),
        "category": fake.word(),
    }
    products.append(product)

    if i in [1, 2]:  # Products 1 and 2 will have variants
        for j in range(1, random.randint(2, 5)):
            variant = {
                "id": i * 10 + j,
                "product_id": i,
                "size": random.choice(["S", "M", "L", "XL"]),
                "color": fake.color_name(),
            }
            variants.append(variant)
```

In [ ]:

```
# Generating sample data for customers and addresses
customers = []
addresses = []

for i in range(1, 11):
    customer = {
        "id": i,
        "name": fake.name(),
        "email": fake.email(),
    }
    customers.append(customer)

    # Each customer will have at least one address
    address = {
        "id": i,
        "customer_id": i,
        "street_address": fake.street_address(),
        "city": fake.city(),
        "postal_code": fake.postcode(),
        "is_current": True,
```

```
    }
    addresses.append(address)
```

In [ ]:

```
# Generating sample data for orders and order items
orders = []
order_items = []

start_date = datetime.date(2022, 1, 1)
end_date = datetime.date(2023, 12, 31)

for i in range(1, 201):
    order_date = fake.date_between_dates(start_date, end_date)
    order = {
        "id": i,
        "customer_id": random.randint(1, 10),
        "order_date": order_date,
        "total_amount": random.uniform(50, 500),
    }
    orders.append(order)

    # Each order will have at least one order item
    order_item = {
        "id": i,
        "order_id": i,
        "variant_id": random.choice(variants)["id"],
        "quantity": random.randint(1, 5),
        "price": random.uniform(20, 100),
    }
    order_items.append(order_item)
```

In [ ]:

```
import pandas as pd

# Sample data (replace this with the generated sample data)
# ... (The same sample data from the previous code)

# Convert to dataframes
products_df = pd.DataFrame(products)
variants_df = pd.DataFrame(variants)
customers_df = pd.DataFrame(customers)
addresses_df = pd.DataFrame(addresses)
orders_df = pd.DataFrame(orders)
order_items_df = pd.DataFrame(order_items)
```

```
Data exported to ecommerce_data.xlsx successfully.
```

In [ ]:

```
products_df.head()
```

Out[ ]:

	id	name	description	category
0	1	trouble	Figure strong already choose left order by.	name
1	2	alone	Community reduce where do our.	fine
2	3	meeting	Little former total particularly give yeah.	receive
3	4	employee	None discover son international too cause stan...	set

	id	name	description	category
4	5	work	Whose understand nature become tax here make.	range

In [ ]:
```
variants_df.head()
```

Out[ ]:

	id	product_id	size	color
0	11	1	S	PeachPuff
1	21	2	XL	Bisque

In [ ]:
```
customers_df.head()
```

Out[ ]:

	id	name	email
0	1	Kristin Young	christophervillanueva@example.com
1	2	Erin Le	villarrealjacqueline@example.net
2	3	Rebecca Wagner	victoria88@example.net
3	4	Brooke Farmer	erichester@example.net
4	5	Kimberly Benjamin	qray@example.org

In [ ]:
```
addresses_df.head()
```

Out[ ]:

	id	customer_id	street_address	city	postal_code	is_current
0	1	1	884 Alejandro Springs	Jacksonview	44817	True
1	2	2	549 Kevin Fork	North Donnaport	75406	True
2	3	3	8883 Eric Divide Apt. 714	Lake Jeffreyfurt	29608	True
3	4	4	0933 Boyer Bridge Suite 425	East Michaelview	82131	True
4	5	5	71773 Perry Run	West Davidborough	03251	True

In [ ]:
```
orders_df.head()
```

Out[ ]:

	id	customer_id	order_date	total_amount
0	1	6	2022-08-12	260.297552
1	2	7	2022-08-10	125.502810
2	3	2	2023-05-06	347.594686
3	4	3	2022-06-19	348.111554
4	5	2	2022-06-06	150.115416

In [ ]:
```
order_items_df.head()
```

Out[ ]:

	id	order_id	variant_id	quantity	price
0	1	1	11	2	38.841535
1	2	2	11	1	67.181983
2	3	3	11	2	38.862350
3	4	4	11	2	30.020153
4	5	5	11	1	72.636401

In [ ]:
```
# Exporting to a single Excel file with different sheets
with pd.ExcelWriter('Jagannath_Ecommerce_Data.xlsx') as writer:
    products_df.to_excel(writer, sheet_name='Products', index=False)
    variants_df.to_excel(writer, sheet_name='Variants', index=False)
    customers_df.to_excel(writer, sheet_name='Customers', index=False)
    addresses_df.to_excel(writer, sheet_name='Addresses', index=False)
    orders_df.to_excel(writer, sheet_name='Orders', index=False)
    order_items_df.to_excel(writer, sheet_name='Order_Items', index=False)

# Print confirmation message
print("Data exported to ecommerce_data.xlsx successfully.")
```

```
Data exported to ecommerce_data.xlsx successfully.
```

In [ ]:
```

```