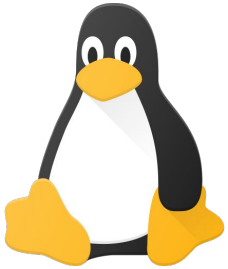
Project 2:Data Analysis

Michael Mengistu, Dane Sperling,
Kenneth Muriel, James Bushek

Tech Stack

- Python
- Pyspark
- Git
- Linux
- Hive
- VS code
- PowerBI
- Hadoop



Project 2: Phase 1 Overview

- Develop a data generator using Python and PySpark to simulate an e-commerce dataset with 10,000 to 15,000 records for trend analysis and visualization.
- Introduce a small percentage of "rogue" that makes up less than 5% of the total data.
- Output the data we generated on to a csv file.
- utilize the agile method and Trello to manage our tasks

Fields (Schema)

Field name	Description
order_id	Order Id
customer_id	Customer Id
customer_name	Customer Name
product_id	Product Id
product_name	Product Name
product_category	Product Category
payment_type	Payment Type (card, Internet Banking, UPI, Wallet)
qty	Quantity ordered
price	Price of the product
datetime	Date and time when order was placed
country	Customer Country
city	Customer City
ecommerce_website_name	Site from where order was placed
payment_txn_id	Payment Transaction Confirmation Id
payment_txn_success	Payment Success or Failure (Y=Success. N=Failed)
failure_reason	Reason for payment failure

Designing Our Data Generator

- Data generated is based off of a Furniture E-Commerce store.
- 20 unique customers
- 22 unique products
- 7 product categories
- 5 website names
- Data range: 1/1/2020 to 1/1/2022
- Added between 1 to 2 percent dirty records at the end.

Products

- Product_Id, Product_Name, Product_Category, & Price are tightly coupled when generating our data.

Product ID	Product Name	Product Category	Price
1	Classic Wooden Bed	Bedroom Furniture	299.99
2	Memory Foam Mattress	Bedroom Furniture	199.99
3	L-Shaped Sofa	Living Room Furniture	499.99
4	Recliner Chair	Living Room Furniture	249.99
5	Oak Dining Table	Dining Room Furniture	399.99
6	Leather Dining Chair	Dining Room Furniture	89.99
7	Wooden Coffee Table	Living Room Furniture	149.99
8	Modular Bookcase	Home Office Furniture	199.99
9	Ergonomic Office Chair	Home Office Furniture	129.99
10	Metal Bed Frame	Bedroom Furniture	149.99
11	Vanity Dresser	Bedroom Furniture	189.99
12	Upholstered Bench	Entryway Furniture	99.99
13	Glass Console Table	Entryway Furniture	129.99
14	TV Stand	Living Room Furniture	89.99
15	Outdoor Patio Set	Outdoor Furniture	299.99
16	Garden Lounge Chair	Outdoor Furniture	79.99
17	Kids Bunk Bed	Kids Furniture	249.99
18	Nursery Rocking Chair	Kids Furniture	119.99
19	Shoe Storage Cabinet	Entryway Furniture	59.99
20	Standing Desk	Home Office Furniture	249.99
21	Blanket	Bedroom Furniture	29.99
22	Hammock	Outdoor Furniture	49.99

Customers

- customer_id, customer_name, country, and city are tightly coupled when generating the data as well.

customer_id	customer_name	country	city
1	Alice Johnson	USA	Phoenix
2	Bob Smith	USA	Los Angeles
3	Catherine Lee	Canada	Toronto
4	Daniel Brown	Australia	Melbourne
5	Eva Garcia	Germany	Cologne
6	Frank Wilson	Canada	Calgary
7	Grace Thompson	Germany	Frankfurt
8	Harry Martinez	Canada	Vancouver
9	Ivy Clark	Australia	Melbourne
10	Jack Lewis	Australia	Perth
11	Karen Young	Germany	Frankfurt
12	Liam Walker	Australia	Adelaide
13	Mona Hall	Germany	Berlin
14	Nathan King	Canada	Calgary
15	Olivia Scott	Canada	Calgary
16	Peter Adams	Australia	Perth
17	Quinn Baker	Australia	Perth
18	Rachel Evans	Germany	Munich
19	Sam Phillips	Canada	Montreal
20	Tina Rodriguez	USA	Chicago

Trends

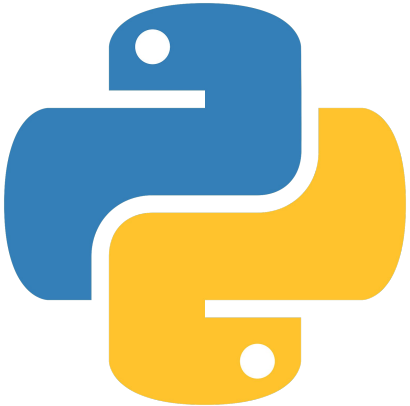
We came up with 4 trends:

1. product_name = “blanket” sells only during december
2. product_name = “hammock” only sells between June 1 and August 31
3. "Chicago", "Vancouver", "Los Angeles", "Berlin" are cities where people like to buy in bulk
4. A good portion of the payment type is done with internet banking

Implementation (version 1)

Version 1 includes:

- data_maker.py
- func.py
- var.py



var.py

Contains important variables for our data_maker.py file:

- Customers: list of customers with customer_id, customer_name, country, and city
- Products: list of products with product_id, product_name, product_category, and price
- product_blanket : list containing blanket information
- product_hammock: list containing hammock information
- payment: list containing payment types
- reason: list containing reasons why payment failed
- success: list containing whether or not a payment was successful
- Websites: list containing the 5 different websites

func.py

Contains functions that do not use pyspark:

- generate_product(prod)
- generate_customers()
- random_date(start, end)
- december_date()
- random_date_in_summer()

data_maker.py

Contains our spark code:

- Created a spark session
- generate_records(num_records, date_generator, product_type, pay_type, order_id)
- Added our trends using the function above
- Defined the schema
- Created our data frame
- Introduced 1 to 2 percent of Rogue Data
- Outputted our data into a CSV

```
def generate_records(num_records, date_generator, product_type, pay_type, order_id):
    data = []
    for i in range(num_records):
        product_id, product_name, product_category, price = generate_product(product_type)
        customer_id, customer_name, country, city = generate_customers()
        payment_txn_success = random.choice(["Y", "N"])
        bulk_mulitplier = check_city(city)
        if pay_type == True:
            payment = "Internet Banking"
        else:
            payment = random.choice(["Card", "Internet Banking", "UPI", "Wallet"])
        record = Row(
            order_id=order_id+i+1,
            customer_id= customer_id,
            customer_name= customer_name,
            product_id=product_id,
            product_name=product_name,
            product_category=product_category,
            payment_type=payment,
            qty=random.randint(1, 5) * bulk_mulitplier,
            price=price,
            datetime=date_generator().strftime("%Y-%m-%d %H:%M:%S"),
            country=country,
            city= city,
            ecommerce_website_name= random.choice(websites),
            payment_txn_id=str(random.randint(1000, 9999)),
            payment_txn_success= payment_txn_success,
            failure_reason= "SUCCESS" if payment_txn_success == "Y" else random.choice(["Insufficient Funds", "Network Error"])
        )
        data.append(record)
    return data
```

Adding Rogue Data

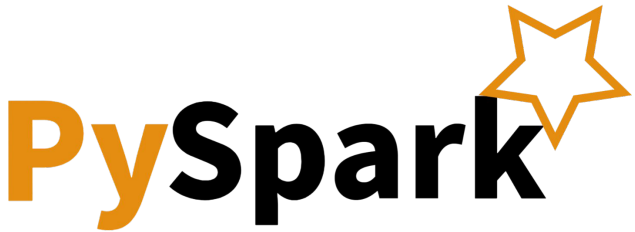
- Added rogue data to product_id and country column.

```
# nullify some rows from product_id
column_to_nullify = "product_id"

# Nullify approximately 1% of rows from product_id
df = df.withColumn(
    column_to_nullify,
    when(rand() < 0.01, lit(None)).otherwise(col(column_to_nullify))
)

# nullify some rows from country
column_to_nullify = "country"

# Nullify approximately 1% of rows from country
df = df.withColumn(
    column_to_nullify,
    when(rand() < 0.01, lit(None)).otherwise(col(column_to_nullify))
)
```

Implementation (version 2)

Version 2 includes: (uses more pyspark then python)

- data_maker_2.py
- var.py



data_maker_2.py

Mainly uses PySpark:

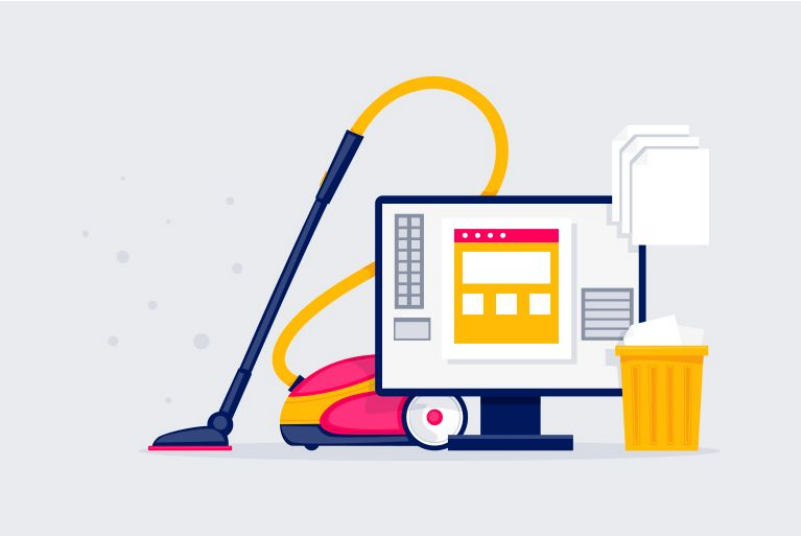
- Creates a spark session
- Multiple data frames were used for products, customers, blanket trends, and hammock trends
- 2 custom user defined functions from pyspark.sql.functions
- Utilizes methods like join, union, range, withColumn, withColumnRenamed in order to manipulate, rename, create columns and rows, merge columns and data frames together to create our dataset.
- Implements the same rogue data generator as data_maker.py
- Outputs our dataset to a csv file

Data_maker_2.py DEMO

Project 2: Phase 2 Overview

- Analyse the dataset
- Find/Clean the rogue data
- Find trends
- Create power BI reports

Cleaning The Data



Before cleaning: 15,000 records

data_cleaner.py:

- Start sparksession
- Import the dataset
- Drop all records with NaN or NULL except in “failure_reason”
- Filter out negative data by qty and price
- Output the cleaned data into a csv

After cleaning: 14,458 records

data_cleaner.py DEMO

Power BI Reports