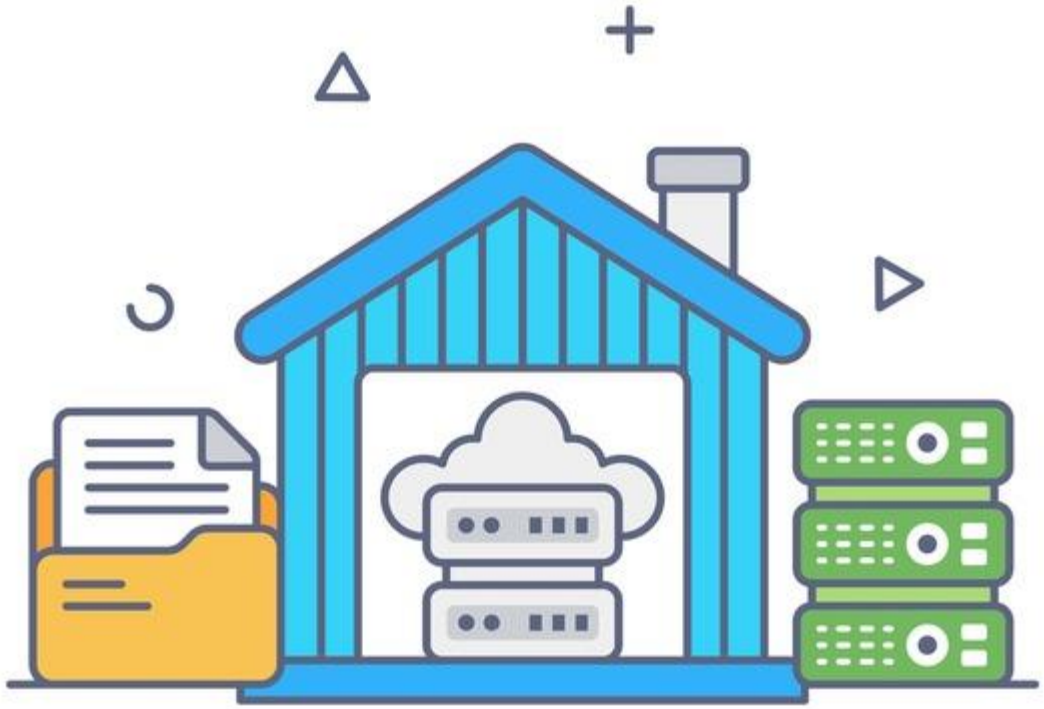

Data Warehouse

Prepare /

- **Student:** Ahmed Wesam Alhayek.
- **Student:** Eid Ahmed Abu Bayd.
- **Student:** Ibrahim Khalil Daoud.

Description

- A simple application for working with retail data for marketing products using Python and SQL that allows company managers to answer questions and display the results, which contributes to understanding the data and the user's understanding, such as knowing which products are selling the most and predicting which products will be in demand in the future.

Sections:

- **First section:** Processing and adjustment.
- **Second section:** Connecting tables to a database and dealing with them through SQL.
- Questions
- Third section: Data Stores and SQL.

The first section is the process of reading the data and collecting it in a folder until it is processed and cleaned to be ready for analysis and reporting

I have five tables:

- **Table I: Sales table**
  - Contains: sales data, products, prices, quantity, date of transaction, and place of sale.
- **Table II: Customers table**
  - Contains: customer ID, gender, age, and payment method.
- **Table III: Branches table**
  - Contains: branch IDs, shopping mall names, and locations.
- **Table IV: Categories table**
  - Contains: category IDs and product categories.
- **Table V: Products table**
  - Contains: product IDs, product names, and associated categories.

First step: Call the libraries and download them if they are not installed.

```
import pandas as pd
import numpy as np
```

[4] ✓ 0.0s

- **Second Step:** Read the tables and store them in variables.

```
products.head()
```

[2] ✓ 0.0s

	product_id	category	product_name	price
0	e735a2b2	might	herself	472
1	c6819930	travel	state	816
2	3cc7a64f	few	election	937
3	ca963ebc	view	Mr	657
4	c096723d	main	forget	423

```
branches.head()
```

[3] ✓ 0.0s

	branch_id	shopping_mall	location	manager_name
0	5ccecd75	Thompson, Sandoval and Wiley	Joeltown	Benjamin Salinas
1	adbdbbf0	Brown-Soto	Port Howard	Gary Gallegos
2	08b65d4d	Johnson, Byrd and Perez	Lake Ronaldhaven	Matthew Harper PhD
3	b44a9204	White and Sons	South Davidburgh	Stacey Figueroa
4	11225458	Young, Baker and Brown	Matthewbury	Anna Johnson

```
categories.head()
```

[6] ✓ 0.0s

	category_id	category	description
0	4d87684d	section	Meeting growth have dinner learn seek sea.
1	ec7b89aa	animal	Really music condition again lay.
2	62802862	per	Particular third reduce strategy simple natura...
3	1a6061ce	everybody	Agree story information statement catch so com...
4	8af88b16	indicate	Forward method central guess detail claim chec...

```
customers.head()
```

[7] ✓ 0.0s

	customer_id	gender	age	payment_method
0	CUS-00001	Female	46	Cash
1	CUS-00002	Female	42	Online
2	CUS-00003	Female	66	Online
3	CUS-00004	Male	62	Cash
4	CUS-00005	Male	65	Online

```
invoices.head()
```

[8] ✓ 0.0s

	invoice_no	customer_id	category	quantity	price	invoice_date	shopping_mall
0	INV-0000001	CUS-00001	long	1	17	2025-01-01	Wilkins and Sons
1	INV-0000002	CUS-00002	group	6	40	2025-01-06	Baker-Booker
2	INV-0000003	CUS-00003	red	9	61	2025-01-11	Gibson and Sons
3	INV-0000004	CUS-00004	Mrs	1	57	2025-01-12	Andrade Group
4	INV-0000005	CUS-00005	customer	10	30	2025-01-03	Nelson, Fry and Campbell

- **Step three:** Print the columns for each table.

```
+ Code   + Markdown

invoices.columns
[10]  ✓ 0.0s

Index(['invoice_no', 'customer_id', 'category', 'quantity', 'price',
       'invoice_date', 'shopping_mall'],
      dtype='object')
```

- **Step four:** Ensure that all columns for each table are unique

```
def unique_value_to_col(data):
    for column in data.columns :
        unique_value = data[column].unique()
        print(f'column {column} is unique value is {unique_value}')

2]  ✓ 0.0s
```

- **Step five:** Create a function that knows three things about the table:
  - 1- Basic information (column names, data type, etc.)
  - 2- Know if the table contains null values.
  - 3- Identify duplicate data

```python
def clean_data (table):
    print('---------- information to data ------------')
    print(table.info())
    print('---------- check null value ------------')
    print(table.isnull().sum())
    print('------------- check duplicated raw ---------------')
    print(table.duplicated().sum())
```

- Step five: Modify the values of the age column in case the values are empty by taking the average values of the age + converting the age values to a data type of an integer value type after making sure that the column does not contain empty values after the modification.

```
cus_data['age'] = cus_data['age'].fillna(cus_data['age'].median())
cus_data.isnull().sum()
```

```
customer_id       0
gender            0
age               0
payment_method    0
dtype: int64
```

- Step six: Convert the date column to a data type of date (day, month, year)

```
sale_data['invoice_date'] = pd.to_datetime(sale_data['invoice_date'], format='%d-%m-%Y')
print(sale_data['invoice_date'].dtypes)
sale_data.info()
```

- **Seventh step:** Save the processing to and download a processed table other than the original table.

```
products.to_csv("products.csv", index=False)
branches.to_csv("branches.csv", index=False)
categories.to_csv("categories.csv", index=False)
customers.to_csv("customers.csv", index=False)
invoices.to_csv("invoices.csv", index=False)
```

✓ 7.6s

Note: At the end of the first section, we now have two tables, each containing cleaned and organized data free of errors or null values.

Second section:

Connecting tables to a database and dealing with them through SQL

- **Initial step:** Call the libraries or install them if they are not installed.

```
import psycopg2
import pandas as pd
from psycopg2 import sql
from psycopg2 import OperationalError
from psycopg2 import sql,OperationalError
from sqlalchemy import create_engine
```

[1] ✓ 5.9s

- **Step 2:** Create a database in Postgres and make sure to connect to it to load the tables' data.

```
try:
    # الاتصال بقاعدة البيانات
    conn = psycopg2.connect(
        dbname="project_DB",
        user="postgres",
        password="123",
        host="localhost",
        port="5432"
    )

    # إنشاء كائن cursor
    cursor = conn.cursor()
```

- Step 3: Create a function that makes sure the tables are in the database connected to the code.

```python
def check_table_exists(cursor, table_name):
    cursor.execute("""
        SELECT EXISTS (
            SELECT 1 FROM information_schema.tables
            WHERE table_schema = 'raw_tables' AND table_name = %s
        );
    """, (table_name,))
    return cursor.fetchone()[0]
```

- Step four: After confirming the connection and the presence of the tables in the database, we can now transfer the data to the tables in the database. And send an error message if we encounter any issues.

```python
# بناء استعلام INSERT مع placeholder
insert_query = sql.SQL("INSERT INTO raw_tables.{} ({}) VALUES ({})").format(
    sql.Identifier(table_name),  # تخصيص اسم الجدول
    sql.SQL(', ').join(map(sql.Identifier, columns)),  # تخصيص الأعمدة
    sql.SQL(', ').join([sql.Placeholder()] * len(columns))  # تخصيص القيم
)

# إدخال البيانات إلى قاعدة البيانات
for row in data.itertuples(index=False, name=None):
    cursor.execute(insert_query, row)
```

- Include a set of user errors in a variable and display them to the user to make it easier for them to resolve the issue

```
except pd.errors.EmptyDataError:
    print(f"Error: The file '{filepath}' is empty")

except FileNotFoundError:
    print(f"Error: The file '{filepath}' was not found")

except OperationalError as e:
    print(f"Operational error while loading data into table '{table_name}': {e}")

except Exception as e:
    print(f"An error occurred while loading data into table '{table_name}': {e}")
```

- Step 5: After all this is done, the function is called to do all the commands at once

```python

if __name__ == "__main__":
    try:
        load_data_to_postgres(r"E:/مخازن بيانات/project2/processed/branches_clean.csv", "branches_clean")
        load_data_to_postgres(r"E:/مخازن بيانات/project2/processed/categories_clean.csv", "categories_clean")
        load_data_to_postgres(r"E:/مخازن بيانات/project2/processed/customers_clean.csv", "customers_clean")
        load_data_to_postgres(r"E:/مخازن بيانات/project2/processed/invoices_clean.csv", "invoices_clean")
        load_data_to_postgres(r"E:/مخازن بيانات/project2/processed/products_clean.csv", "products_clean")


```

Note: The last step of the second section is to enter the data into the tables in the database and the analysis, statistics, and questions will be done using sql.

The following image shows that the tables have been successfully added to the database

```
^

[Done] exited with code=0 in 1.34 seconds

[Running] python -u "e:\مخازن بيانات\project2\final_lap.py"
Data loaded successfully!
Data loaded successfully!
Data loaded successfully!
e:\\u0645\u062e\u0627\u0632\u0646 \u0628\u064a\u0627\u0646\u0627\u062a\project2\final_lap.py:37: DtypeWarning: Columns (2,3,4) have mixed types. Specify dtype option
on import or set low_memory=False.
  data = pd.read_csv(filepath)
Data loaded successfully!
Data loaded successfully!

[Done] exited with code=0 in 15.755 seconds
```

Third section: Data Stores and SQL

- **First step:** Since I have two tables, another specialized schema will be created in the data warehouses to create dimension tables and fact tables to facilitate analysis and reporting, and the star chart will be adopted for this.

```sql

-- create new SCHEMA
CREATE SCHEMA dw;
```

Second Step: Create dimension and fact tables

- Fact table:

```sql
-- Creating Invoices Fact Table in DW
CREATE TABLE IF NOT EXISTS dw.invoices_fact (
    invoice_no VARCHAR(255) PRIMARY KEY,   -- Invoice number
    customer_id VARCHAR(255),              -- Customer ID (foreign key to customers_dimension)
    product_id VARCHAR(255),               -- Product ID (foreign key to products_dimension)
    branch_id VARCHAR(255),                -- Branch ID (foreign key to branches_dimension)
    category_id VARCHAR(255),              -- Category ID (foreign key to categories_dimension)
    category VARCHAR(255),                 -- Product's category
    quantity INT,                          -- Quantity of product sold
    price DECIMAL(10, 2),                  -- Price of the product
    invoice_date DATE,                     -- Date of the invoice
    shopping_mall VARCHAR(255)            -- Shopping mall where the sale occurred
);
```

- Dimension table:

```sql
-- Creating Branches Dimension Table in DW
CREATE TABLE IF NOT EXISTS dw.branches_dimension (
    branch_id VARCHAR(255) PRIMARY KEY,  -- Unique ID for each branch
    shopping_mall VARCHAR(255),          -- Shopping mall where the branch is located
    location VARCHAR(255),               -- Branch's location
    manager_name VARCHAR(255)            -- Manager of the branch
);
```

Dimension table:

```sql
-- Creating Products Dimension Table in DW
CREATE TABLE IF NOT EXISTS dw.products_dimension (
    product_id VARCHAR(255) PRIMARY KEY,   -- Unique ID for each product
    category VARCHAR(255),                 -- Product's category
    product_name VARCHAR(255),             -- Name of the product
    price DECIMAL(10, 2)                   -- Price of the product
);
```

```sql
-- Creating Customers Dimension Table in DW
CREATE TABLE IF NOT EXISTS dw.customers_dimension (
    customer_id VARCHAR(255) PRIMARY KEY,  -- Unique ID for each customer
    gender VARCHAR(50),                    -- Gender of the customer
    age INT,                               -- Age of the customer
    payment_method VARCHAR(100)            -- Method of payment used by the customer
);
```

```sql
-- Creating Categories Dimension Table in DW
CREATE TABLE IF NOT EXISTS dw.categories_dimension (
    category_id VARCHAR(255) PRIMARY KEY,  -- Unique ID for each category
    category VARCHAR(255),                 -- Name of the category
    description TEXT                       -- Description of the category
);
```

Step three: Add the data to the tables and make sure it is correct.

```sql

-- إضافة البيانات من branches_clean إلى dw.branches_dimension
INSERT INTO dw.branches_dimension (branch_id, shopping_mall, location, manager_name)
SELECT branch_id, shopping_mall, location, manager_name
FROM raw_tables.branches_clean;

-- إضافة البيانات من categories_clean إلى dw.categories_dimension
INSERT INTO dw.categories_dimension (category_id, category, description)
SELECT category_id, category, description
FROM raw_tables.categories_clean;

-- إضافة البيانات من customers_clean إلى dw.customers_dimension
INSERT INTO dw.customers_dimension (customer_id, gender, age, payment_method)
SELECT customer_id, gender, age, payment_method
FROM raw_tables.customers_clean;

-- إضافة البيانات من products_clean إلى dw.products_dimension
INSERT INTO dw.products_dimension (product_id, category, product_name, price)
SELECT product_id, category, product_name, price
FROM raw_tables.products_clean;

-- إضافة البيانات من invoices_clean إلى dw.invoices_fact
INSERT INTO dw.invoices_fact (invoice_no, customer_id, product_id, branch_id, category_id, category, quantity, price, invoice_date, shopping_mall)
SELECT invoice_no, customer_id, product_id, branch_id, category_id, category, quantity, price, invoice_date, shopping_mall
FROM raw_tables.invoices_clean;


```

Step four: Submit questions and answer them for the company or a specialized person for that matter.

- 1. Analyze the number of branches by shopping mall and location
- 2. Analyze the number of products in each category
- 3. Analyze customer demographics (average age and total customers by gender)
- 4. Analyze total sales for each product
- 5. Analyze total revenue for each category

Q1:

```sql
-- 1. Analyze the number of branches by shopping mall and location
-- This query shows how many branches are located in each shopping mall and location.
SELECT branch_id, shopping_mall, location, COUNT(*) AS num_branches
FROM dw.branches_dimension
GROUP BY branch_id, shopping_mall, location;
```

Data Output Messages Notifications

	branch_id [PK] character varying (255)	shopping_mall character varying (255)	location character varying (255)	num_branches bigint
1	58a0f5e7	Delgado-George	Phillipston	1
2	4dbfef86	Cruz and Sons	Paulchester	1
3	bbe3a33a	Robinson, Woods and Keith	Markberg	1
4	928ba065	Smith-Simon	Anthonyburgh	1
5	ba148b98	Mcdonald, Rivera and Mendoza	Port Kellytown	1
6	13ea8a67	Richardson, Moody and Nguyen	Bryanmouth	1
7	f0318df6	Mcmillan-Guerra	East Paulfurt	1

Q2.

```sql
-- 2. Analyze the number of products in each category
-- This query counts how many products are available in each category.
SELECT category_id, category, COUNT(*) AS num_products
FROM dw.categories_dimension
GROUP BY category_id, category;
```

Data Output Messages Notifications

	category_id [PK] character varying (255)	category character varying (255)	num_products bigint
1	510c1f8d	offer	1
2	c1a05487	society	1
3	3a0a5768	woman	1
4	0ea03ba3	power	1
5	420706a1	computer	1
6	99d00d36	city	1
7	bf6b0c75	answer	1

Q3.

```sql
-- 3. Analyze customer demographics (average age and total customers by gender)
-- This query calculates the average age and total number of customers, grouped by gender.
SELECT customer_id, gender, AVG(age) AS avg_age, COUNT(*) AS total_customers
FROM dw.customers_dimension
GROUP BY customer_id, gender;
```

Data Output Messages Notifications

	customer_id [PK] character varying (255)	gender character varying (50)	avg_age numeric	total_customers bigint
1	CUS-00001	Female	22.0000000000000000	1
2	CUS-00002	Female	40.0000000000000000	1
3	CUS-00003	Male	27.0000000000000000	1
4	CUS-00004	Female	63.0000000000000000	1
5	CUS-00005	Female	24.0000000000000000	1
6	CUS-00006	Female	30.0000000000000000	1
7	CUS-00007	Female	67.0000000000000000	1

Q4.

```sql
-- 4. Analyze total sales for each product
-- This query calculates the total quantity sold and total revenue for each product.
SELECT product_id, SUM(quantity) AS total_quantity, SUM(price * quantity) AS total_sales
FROM dw.invoices_fact
GROUP BY product_id;
```

Data Output | Messages | Notifications

	product_id character varying (255)	total_quantity bigint	total_sales numeric
1	f0e3cf6e	2	172.00
2	bf6b9d15	2	116.00
3	4ef29beb	4	380.00
4	32fc82c3	7	462.00
5	8405db4c	4	384.00
6	36fde414	2	90.00
7	a4ecb390	3	75.00

Q5.

```sql
-- 5. Analyze total revenue for each category
-- This query sums up the revenue for each category, showing which categories have the highest sales.
SELECT category, SUM(price * quantity) AS total_revenue
FROM dw.invoices_fact
GROUP BY category;
```

Data Output Messages Notifications

	category character varying (255)	total_revenue numeric
1	answer	1545570.00
2	city	1515663.00
3	guess	1541888.00
4	computer	1508528.00
5	rest	1502957.00
6	exist	1480246.00
7	woman	1513817.00

External learning resources and references/

Reference	URL	Notes
To learn SQL	https://harmash.com/tutorials/sql/overview	web
To make the diagram	https://miro.com/app/dashboard/	web
To write SQL commands and database connectivity	PostgreSQL	app
To write code in Python	Jupyter	app

تم بحمد الله