Task-4 E-Commerce_Project

📖 Project Overview

This project focuses on analyzing and managing data for an e-commerce platform. The objective is to build insights into customer behavior, product performance, and operational efficiency. The project uses SQL for data handling, Python (via Jupyter Notebook) for ETL processes, and various datasets to generate actionable insights.

🛠️ Features and Highlights

- **SQL Scripting: Comprehensive SQL queries for data creation, insertion, validation, and integrity checks.**
- **ETL Pipeline: A Jupyter Notebook for extracting, transforming, and loading data into analytical workflows.**
- **Key Performance Indicators (KPIs):**
  - **Customer Lifetime Value Analysis**
  - **Product Inventory Reports**
- **Datasets: Realistic datasets covering orders, payments, products, reviews, and users.**
- **Reports: Final outputs to guide strategic decision-making.**

📁 Directory Structure

E-Commerce Project/

├── Data/ # Raw datasets

│ ├── orders_data.csv

│ ├── payments_data.xlsx

│ ├── products_data.csv

│ ├── reviews_data.xlsx

│ └── users_data.csv

├── Reports/ # Analysis outputs

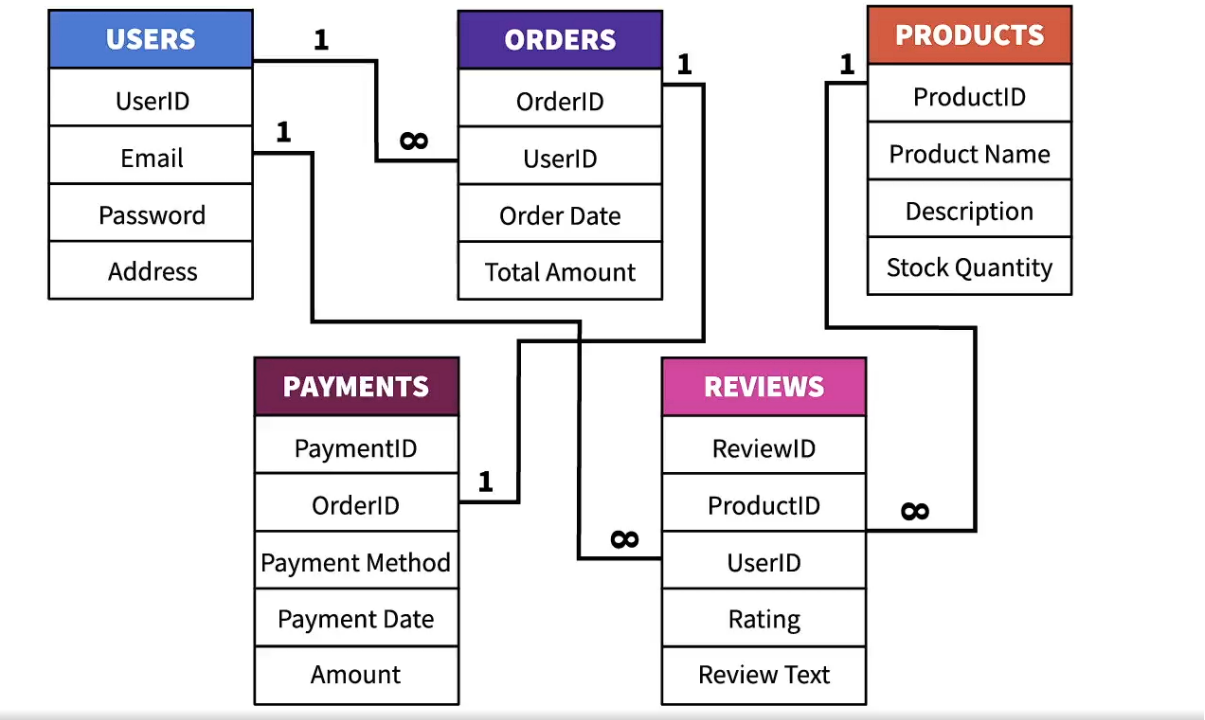
│ ├── Customer_Lifetime_Value.csv

│ └── Product_Inventory_Report.csv

├── SQL Scripts/ # SQL queries for database management

```
│   ├── Data_Efficiency_Queries.sql
│   ├── Data_Insert_Queries.sql
│   ├── Data_Integrity_Queries.sql
│   ├── Data_Validation_Queries.sql
│   ├── Date_Insert_Queries.sql
│   ├── KPI_Analysis_Queries.sql
│   └── Table_Creation_Queries.sql
├── ETL_ECommerce_Project.ipynb # Python-based ETL pipeline
└── README.md                   # Project documentation
```

📊 Database ERD Diagram



⚙️ Tools and Technologies

- **SQL: Used for database creation, validation, and KPI queries.**
- **Python: Jupyter Notebook for ETL processes.**
- **Data Visualization: Insights and outputs visualized using tabular reports.**
- **File Formats: Data stored in CSV and Excel formats for structured analysis.**

ETL Pipeline - Key Component of the Project 🚀

Overview

The ETL (Extract, Transform, Load) pipeline is a critical part of this project. It processes raw e-commerce data into clean, structured formats, ready for analysis. The ETL pipeline is implemented in Python within the ETL_ECommerce_Project.ipynb notebook.

Modules and Process

1. **Extract:**
    - **Data is sourced from multiple files, including:**
        - **CSV files: orders_data.csv, products_data.csv, users_data.csv**
        - **Excel files: payments_data.xlsx, reviews_data.xlsx**
    - **Libraries Used: pandas, openpyxl**
2. **Transform:**
    - **Data Cleaning:**
        - **Handled missing values, duplicate entries, and inconsistent formats.**
    - **Data Enrichment:**
        - **Added calculated fields such as total payment amounts, product ratings averages, etc.**
    - **Normalization:**
        - **Ensured data is structured for relational database compatibility.**
    - **Libraries Used: pandas, numpy**
3. **Load:**
    - **Transformed data is loaded into a structured database using SQL scripts or stored as processed CSVs for analysis.**
    - **Integration with sqlite3 or any preferred database can be configured.**
    - **Libraries Used: sqlalchemy, sqlite3**

What We've Done

- **Unified raw datasets from multiple sources into a cohesive structure.**
- **Automated cleaning processes to handle thousands of data points efficiently.**

- **Generated a dataset ready for SQL analysis and KPI computations.**
- **Documented all steps in a clear and reproducible notebook for transparency and scalability.**

This ETL process is designed for flexibility and adaptability, making it a robust framework for handling e-commerce data workflows!

🚀 What We Accomplished

- **Database Management: Created and maintained normalized databases using SQL.**
- **ETL Processes: Automated the transformation and loading of datasets for analysis.**
- **Insights Generation:**
  - **Customer Lifetime Value report to understand customer profitability.**
  - **Product Inventory report to track inventory performance.**
- **Data Integrity: Ensured high data quality through validation scripts.**

🔍 How to Use

1. **Clone the repository:**

git clone https://github.com/YourUsername/E-Commerce-Data-Analysis.git

2. **Navigate to the project directory:**

cd E-Commerce-Data-Analysis

3. **Set up a database using the provided SQL scripts.**
4. **Run the ETL process using ETL_ECommerce_Project.ipynb.**
5. **Analyze the final reports in the Reports/ folder.**

📊 Example Use Cases

1. **Customer Insights:**
   - **Identify high-value customers and tailor marketing strategies.**
2. **Inventory Management:**
   - **Streamline product restocking based on inventory trends.**
3. **Data Quality:**
   - **Use validation scripts to ensure consistency and reliability of datasets.**

Data_Efficiency_Queries.sql

Code:

```
-- USING INDEX

Create index category_idx on Products(Category);


-- USING THE SAME QUERY AGAIN


EXPLAIN Select ProductName,Price

from ecommerce_1.products_data

where Category='Electronics';
```

Output:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	products_data	NULL	ALL	NULL	NULL	NULL	NULL	60	10.00	Using

Data_Insert_Queries.sql

Code:

```
INSERT INTO PRODUCTS (ProductName,Description,Price,StockQuantity,Category)

VALUES

('Apple Iphone 16','Latest Smartphone by Apple','699.99',50,'Electronics'),

('Nike Jordans','New Launched Sneakers by Nike','99.99',100,'Sneakers'),

('Boss Bass 20','Portable Speaker with Excellent Sound Quality','150.00',200,'Electronics');


SELECT * FROM PRODUCTS;


INSERT INTO USERS(UserName,Email,Password,Address)

VALUES
```

('ankit1222000','ankit.bhatia1220@gmail.com','Ankit@1222000','1008, Greek Row Dr Apt 101'),

('bhavesh6599','bwadhwa0387@gmail.com','Bhavesh@6599','South Campus Apt 10');

select * from users;

insert into orders(UserID,ProductID,OrderDate,OrderQuantity,TotalAmount)

values

(1,3,'2024-11-24',3,450),

(2,1,'2024-11-24',1,699.99);

select * from orders;

insert into reviews(UserID,ProductID,Rating,ReviewText)

values

(1,3,5,'Amazing Speakers, very satisfied'),

(2,1,4,'Great Phone, could be less expensive');

select * from reviews;

insert into Payments(orderId,PaymentMethod,PaymentDate,Amount)

values

(1,'Cash','2024-11-24',450),

(2,'Credit Card','2024-11-24',699.99);

select * from payments;

output:

ï»¿OrderID	PaymentMethod	PaymentDate	Amount
1001	InvalidMethod		
1002	Credit Card	2024-08-02 00:00:00	753.48
1003	Credit Card	2024-08-03 00:00:00	978.27
1004	InvalidMethod	2024-08-04 00:00:00	564.67
1005	Credit Card	2024-08-05 00:00:00	390.66
1006	Credit Card		815.67
1007	InvalidMethod	2024-08-07 00:00:00	343.75
1008	Credit Card	2024-08-08 00:00:00	495.07
1009	Credit Card	2024-08-09 00:00:00	170.61

Data_Integrity_Queries.sql

Code:

```
use ecommerce_1;


ALTER TABLE ORDERS

ADD CONSTRAINT

chk_user

foreign key (UserID) references Users(UserID)

ON DELETE

CASCADE;

START TRANSACTION;

INSERT INTO ORDERS(ORDERID,USERID,OrderDate,TotalAmount)

VALUES(1009,70,CURDATE(),"ANKIT");

COMMIT;

rollback;

SELECT * FROM ecommerce_1.orders_data;
```

Output:

UserID	OrderDate	TotalAmount	Email
8	InvalidDate		user8@example.com
4	2024-08-02 00:00:00	365.0	user4@example.com
1	2024-08-03 00:00:00	446.59	user1@example.com
8	2024-08-04 00:00:00	866.02	user8@example.com
4	2024-08-05 00:00:00	385.23	user4@example.com
6	2024-08-06 00:00:00	252.54	user6@example.com
8	2024-08-07 00:00:00	601.12	user8@example.com
4	InvalidDate	942.54	user4@example.com
3	2024-08-09 00:00:00	726.43	user3@example.com

KPI_Analysis_Queries.sql

Code:

```
use ecommerce_1;



-- SALES ANALYSIS QUERIES
```

```
-- this query calculates the total number of sales of all the products

SELECT

SUM(TotalAmount) AS TotalSales

FROM

ecommerce_1.orders_data;


select MONTHNAME(OrderDate) as MONTH, -- Alternatively can also use
Date_Format(OrderDate,"%M")

SUM(TotalAmount) AS TotalSales

from orders_data

group by

MONTH;



-- ANALYZING CUSTOMER BEHAVIOUR


-- TOP CUSTOMERS

-- THIS QUERY WILL IDENTIFY THE TOP 5 CUSTOMES WHO SPENT THE MOST ON THE
PLATFORM


SELECT

users.UserName, users.Email, users.Address,SUM(orders.TotalAmount) as TotalSpent

from users_data

join

orders on

users.UserID=orders.UserID

group by

users.UserID

order by

TotalSpent DESC

LIMIT 5;
```

```
-- 2. CUSTOMER LIFETIME VALUE (CLV)
SELECT
users.UserName as User, SUM(orders.TotalAmount) as LifeTimeValue
from users_data
join
orders on
users.UserID=orders.UserID
group by
users.UserID
order by
LifeTimeValue;
-- AVERAGE PRICE BY PRODUCT CATEGORY
SELECT Category,Round(Avg(Price),2) as "Average Product Price" FROM products_data
group by Category
order by "Average Product Price";

-- TOTAL INVENTORY BY EACH PRODUCT CATEGORY
SELECT Category,SUM(StockQuantity*Price) as Inventory from products_data
group by Category
order by Inventory;
```

Output:

Category	Inventory
Apparatus	122308.67
Machines	142341.78000000003
Contraptions	150368.57
Instruments	170996.03
Devices	176254.52
Gadgets	176577.6
Appliances	201028.66000000003
Tools	204731.13
Widgets	255834.48

Table_Creation_Queries.sql

Code:

```
use ecommerce_1;

CREATE TABLE PRODUCTS(
ProductID INT AUTO_INCREMENT PRIMARY KEY,
ProductName varchar(255) not null,
Description TEXT,
Price DECIMAL(10,2) not null,
StockQuantity INT not null,
Category varchar(100),
DateAdded Date NOT NULL DEFAULT (CURDATE())
);

DESCRIBE products_data;



CREATE TABLE USERS(
UserID INT auto_increment PRIMARY KEY,
UserName varchar(255) not null,
Email varchar(255) not null unique,
Password varchar(255) not null,
Address varchar(255) not null,
DateRegistered Date not null DEFAULT (CurDate())
);

Describe users_data;
```

```
CREATE TABLE ORDERS(
OrderID INT auto_increment primary key,
UserID int not null,
OrderDate date not null,
TotalAmount Decimal(10,1) not null,
foreign key(UserID) references Users(UserId)
);


describe orders_data;

CREATE TABLE REVIEWS(
ReviewID INT auto_increment primary key,
UserID int not null,
ProductID int not null,
Rating int not null,
ReviewText TEXT,
foreign key(UserID) references Users(UserId),
foreign key(ProductID) references Products(ProductId),
CHECK (Rating>=1 and Rating<=5)
);

Describe reviews_data;


CREATE TABLE Payments(
PaymentID int auto_increment primary key,
OrderID int not null,
PaymentMethod Varchar(50),
PaymentDate Date,
Amount Decimal (10,1),
```

foreign key (OrderID) references Orders(OrderID)

);

describe payments_data;

output:

	Field	Type	Null	Key	Default	Extra
▶	ï»¿OrderID	int	YES		NULL	
	PaymentMethod	text	YES		NULL	
	PaymentDate	text	YES		NULL	
	Amount	text	YES		NULL	