Task 4

CREATE DATABASE IF NOT EXISTS olist;  
USE olist;  
  
CREATE TABLE IF NOT EXISTS customers (  
 customer\_id VARCHAR(50) PRIMARY KEY,  
 customer\_unique\_id VARCHAR(50),  
 customer\_zip\_code\_prefix INT,  
 customer\_city VARCHAR(50),  
 customer\_state VARCHAR(10)  
);  
  
CREATE TABLE IF NOT EXISTS order\_items (  
 order\_id VARCHAR(50),  
 order\_item\_id INT,  
 product\_id VARCHAR(50),  
 seller\_id VARCHAR(50),  
 shipping\_limit\_date DATETIME,  
 price FLOAT,  
 freight\_value FLOAT  
);  
  
CREATE TABLE IF NOT EXISTS order\_payments (  
 order\_id VARCHAR(50),  
 payment\_sequential INT,  
 payment\_type VARCHAR(30),  
 payment\_installments INT,  
 payment\_value FLOAT  
);  
  
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/olist\_customers\_dataset.csv'  
INTO TABLE customers  
FIELDS TERMINATED BY ','   
ENCLOSED BY '"'  
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;  
  
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/olist\_order\_items\_dataset.csv'  
INTO TABLE order\_items  
FIELDS TERMINATED BY ','   
ENCLOSED BY '"'  
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;  
  
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/olist\_order\_payments\_dataset.csv'  
INTO TABLE order\_payments  
FIELDS TERMINATED BY ','   
ENCLOSED BY '"'  
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;

# 2. Data Analysis Queries

This section includes multiple SQL queries used to extract insights from the dataset.  
Queries include order counts by state, total and top payments, most expensive items, payment type distribution, and view creation.

## Orders by State

SELECT customer\_state, COUNT(\*) AS order\_count  
FROM customers  
GROUP BY customer\_state  
ORDER BY order\_count DESC;

## Show Tables

SHOW TABLES;

## Total Payment per Order

SELECT order\_id, SUM(payment\_value) AS total\_payment  
FROM order\_payments  
GROUP BY order\_id;

## Top 5 Expensive Orders

SELECT order\_id, SUM(payment\_value) AS total\_payment  
FROM order\_payments  
GROUP BY order\_id  
ORDER BY total\_payment DESC  
LIMIT 5;

## Top 5 Most Expensive Order Items

SELECT order\_id, product\_id, price  
FROM order\_items  
ORDER BY price DESC  
LIMIT 5;

## Join: Top 5 Most Expensive Orders by Payment + Items

SELECT   
 oi.order\_id,  
 ROUND(SUM(oi.price), 2) AS total\_item\_price,  
 ROUND(SUM(p.payment\_value), 2) AS total\_payment  
FROM order\_items oi  
JOIN order\_payments p ON oi.order\_id = p.order\_id  
GROUP BY oi.order\_id  
ORDER BY total\_payment DESC  
LIMIT 5;

## Payments by Type

SELECT payment\_type, COUNT(\*) AS count, ROUND(SUM(payment\_value), 2) AS total  
FROM order\_payments  
GROUP BY payment\_type;

## Create View for High-Value Orders

CREATE VIEW high\_value\_orders AS  
SELECT order\_id, SUM(payment\_value) AS total\_payment  
FROM order\_payments  
GROUP BY order\_id  
HAVING total\_payment > 500;