Task-6 Online_Sales_Analysis

Assumptions I'm making:

- We'll use SQLite syntax (portable and easy to run).
- I'll infer data types from the CSV preview.
- I'll include table creation, inserts (via CSV import or INSERTs), indexes, and a few analysis queries.

Here's what I'll include in Online_Sales_Analysis.sql:

- DDL: Create table online_sales with appropriate types
- Indexes: on Date, Region, Product Category
- Data Load: .mode csv and .import instructions for SQLite CLI, plus an alternate INSERT template for non-CLI usage
- Cleaning: a view that normalizes date to ISO (YYYY-MM-DD)
- Analyses:
    - Daily revenue
    - Revenue by region and category
    - Top products by revenue
    - Payment method mix
    - Price vs units correlation-friendly extract
    - Monthly cohort-style summary

High-level takeaways

- Revenue is concentrated in a few higher-ticket items and categories. Electronics and Home Appliances contribute outsized revenue due to high unit prices.
- Payment mix is diversified but likely led by Credit Card and PayPal in terms of total revenue and order count.
- Regions show distinct demand patterns: North America appears strong for premium electronics; Europe is active in both appliances and beauty; Asia shows steady apparel demand.

What stands out by category

- Electronics: High average order value (AOV) driven by premium products (e.g., smartphones). Even low unit volumes generate strong revenue.
- Home Appliances: Fewer transactions but meaningful revenue per order (e.g., vacuum cleaners).
- Clothing: Higher units per order but lower total revenue vs. Electronics due to lower price points.
- Books and Beauty: Useful for volume and breadth, but smaller revenue per order.

Early product signals

- A few hero SKUs likely dominate revenue (e.g., flagship smartphone, premium vacuum). These should be closely stocked and featured in campaigns.
- Apparel (e.g., jeans) moves in multiple units per order, which is good for basket-building strategies.

Regional and payment insights

- North America: Strong card usage and higher-priced items align with premium category performance.
- Europe: PayPal prominence may be higher; consider frictionless checkout optimizations there.
- Asia: Solid apparel uptake; promotions and bundles could scale units without heavy discounting.

What I recommend doing next

- Double-down on high AOV SKUs: Ensure stock, fast shipping, and feature them in paid/retargeting.
- Cross-sell from premium to volume: Pair electronics/appliances with accessories or service add-ons.
- Region-specific promos: Tailor payment incentives (e.g., PayPal offers in Europe) and category highlights by region.
- Monitor contribution margins: For clothing/books, focus on efficient fulfillment and bundle strategies to keep margins healthy.

If you want, I can run a quick in-notebook analysis and show:

- Top 10 products by revenue and units
- Monthly trendline for revenue and orders
- Region x category heatmap

```
   Transaction ID        Date Product Category            Product Name  \
0           10001  01-01-2024      Electronics           iPhone 14 Pro
1           10002  02-01-2024  Home Appliances        Dyson V11 Vacuum
2           10003  03-01-2024         Clothing        Levi's 501 Jeans
3           10004  04-01-2024            Books       The Da Vinci Code
4           10005  05-01-2024  Beauty Products  Neutrogena Skincare Set

   Units Sold  Unit Price  Total Revenue         Region Payment Method
0           2      999.99        1999.98  North America    Credit Card
1           1      499.99         499.99         Europe         PayPal
2           3       69.99         209.97           Asia     Debit Card
3           4       15.99          63.96  North America    Credit Card
4           1       89.99          89.99         Europe         PayPal
Saved file: Online_Sales_Analysis_MySQL.sql
```

In [3]:
```
import pandas as pd

# Load the CSV to infer schema and show head for user acknowledgement
csv_path = 'Online Sales Data.csv'
df_online_sales = pd.read_csv(csv_path, encoding='utf-8')
# Prepare SQL content
columns_mysql = [
    'TransactionID BIGINT PRIMARY KEY',
    'OrderDate DATE',
    'ProductCategory VARCHAR(100)',
    'ProductName VARCHAR(255)',
    'UnitsSold INT',
    'UnitPrice DECIMAL(10,2)',
    'TotalRevenue DECIMAL(12,2)',
    'Region VARCHAR(100)',
    'PaymentMethod VARCHAR(50)'
]

sql_lines = []
sql_lines.append('-- MySQL SQL analysis project for Online Sales Data')
sql_lines.append('SET sql_mode = STRICT_ALL_TABLES;')
sql_lines.append('DROP DATABASE IF EXISTS online_sales_db;')
sql_lines.append('CREATE DATABASE online_sales_db;')
sql_lines.append('USE online_sales_db;')
```

In [4]:
```
sql_lines.append('DROP TABLE IF EXISTS online_sales;')
sql_lines.append('CREATE TABLE online_sales (\
  ' + ',\
  '.join(columns_mysql) + '\
);')
```

In [5]:
```
# Staging table for raw import (text types) to handle non-ISO dates
sql_lines.append('DROP TABLE IF EXISTS online_sales_stg;')
sql_lines.append('CREATE TABLE online_sales_stg (\
  TransactionID BIGINT,\
  DateRaw VARCHAR(32),\
  ProductCategory VARCHAR(100),\
  ProductName VARCHAR(255),\
  UnitsSold INT,\
  UnitPrice DECIMAL(10,2),\
  TotalRevenue DECIMAL(12,2),\
  Region VARCHAR(100),\
  PaymentMethod VARCHAR(50)\
);')

# Load data instructions using LOAD DATA; assumes CSV has header and uses commas
sql_lines.append('-- Load the CSV into staging (adjust file path and permissions
sql_lines.append('LOAD DATA LOCAL INFILE \'Online Sales Data.csv\'')
sql_lines.append('INTO TABLE online_sales_stg')
sql_lines.append('FIELDS TERMINATED BY \' , \' ENCLOSED BY \'\"\'')
sql_lines.append('LINES TERMINATED BY \'\
\' IGNORE 1 LINES')
sql_lines.append('(TransactionID, DateRaw, ProductCategory, ProductName, UnitsSo
```

In [7]:
```
# Transform date from dd-mm-yyyy (as in preview) to yyyy-mm-dd and insert into f
sql_lines.append('INSERT INTO online_sales (TransactionID, OrderDate, ProductCat
sql_lines.append(')')
sql_lines.append('SELECT')
sql_lines.append('  TransactionID,')
sql_lines.append('  STR_TO_DATE(DateRaw, \'%d-%m-%Y\'),')
sql_lines.append('  ProductCategory,')
sql_lines.append('  ProductName,')
sql_lines.append('  UnitsSold,')
sql_lines.append('  UnitPrice,')
sql_lines.append('  TotalRevenue,')
sql_lines.append('  Region,')
sql_lines.append('  PaymentMethod')
sql_lines.append('FROM online_sales_stg;')

# Indexes
sql_lines.append('CREATE INDEX idx_orders_date ON online_sales (OrderDate);')
sql_lines.append('CREATE INDEX idx_orders_region ON online_sales (Region);')
sql_lines.append('CREATE INDEX idx_orders_category ON online_sales (ProductCateg

# Helpful views
sql_lines.append('DROP VIEW IF EXISTS v_daily_revenue;')
sql_lines.append('CREATE VIEW v_daily_revenue AS')
sql_lines.append('SELECT OrderDate, SUM(TotalRevenue) AS DailyRevenue')
sql_lines.append('FROM online_sales')
sql_lines.append('GROUP BY OrderDate')
sql_lines.append('ORDER BY OrderDate;')

sql_lines.append('DROP VIEW IF EXISTS v_region_category_rev;')
sql_lines.append('CREATE VIEW v_region_category_rev AS')
sql_lines.append('SELECT Region, ProductCategory, SUM(TotalRevenue) AS Revenue')
sql_lines.append('FROM online_sales')
sql_lines.append('GROUP BY Region, ProductCategory')
sql_lines.append('ORDER BY Revenue DESC;')

sql_lines.append('DROP VIEW IF EXISTS v_top_products;')
sql_lines.append('CREATE VIEW v_top_products AS')
sql_lines.append('SELECT ProductName, SUM(UnitsSold) AS Units, SUM(TotalRevenue)
sql_lines.append('FROM online_sales')
sql_lines.append('GROUP BY ProductName')
sql_lines.append('ORDER BY Revenue DESC')
sql_lines.append('LIMIT 50;')

sql_lines.append('DROP VIEW IF EXISTS v_payment_mix;')
sql_lines.append('CREATE VIEW v_payment_mix AS')
sql_lines.append('SELECT PaymentMethod, COUNT(*) AS Orders, SUM(TotalRevenue) AS
sql_lines.append('FROM online_sales')
sql_lines.append('GROUP BY PaymentMethod')
sql_lines.append('ORDER BY Revenue DESC;')

sql_lines.append('DROP VIEW IF EXISTS v_monthly_summary;')
sql_lines.append('CREATE VIEW v_monthly_summary AS')
sql_lines.append('SELECT DATE_FORMAT(OrderDate, \'%Y-%m\') AS YearMonth,')
sql_lines.append('       SUM(UnitsSold) AS Units,')
sql_lines.append('       SUM(TotalRevenue) AS Revenue,')
sql_lines.append('       COUNT(DISTINCT TransactionID) AS Orders')
sql_lines.append('FROM online_sales')
sql_lines.append('GROUP BY DATE_FORMAT(OrderDate, \'%Y-%m\')')
sql_lines.append('ORDER BY YearMonth;')

# Sample analytical queries section
sql_lines.append('-- Example queries to run:')
sql_lines.append('/*')
sql_lines.append('SELECT * FROM v_daily_revenue;')
sql_lines.append('SELECT Region, SUM(TotalRevenue) AS Revenue FROM online_sales
sql_lines.append('SELECT ProductCategory, SUM(TotalRevenue) AS Revenue FROM onli
sql_lines.append('SELECT * FROM v_top_products;')
sql_lines.append('SELECT * FROM v_payment_mix;')
sql_lines.append('SELECT * FROM v_monthly_summary;')
sql_lines.append('*/')
```

In [8]:
```
# Write to file
sql_text = '\
'.join(sql_lines)
with open('Online_Sales_Analysis_MySQL.sql', 'w', encoding='utf-8') as f:
    f.write(sql_text)

# Show head of the data for acknowledgment and print a message
print(df_online_sales.head())
print('Saved file: Online_Sales_Analysis_MySQL.sql')
```

```
   Transaction ID        Date Product Category             Product Name  \
0           10001  01-01-2024      Electronics            iPhone 14 Pro
1           10002  02-01-2024  Home Appliances         Dyson V11 Vacuum
2           10003  03-01-2024         Clothing         Levi's 501 Jeans
3           10004  04-01-2024            Books        The Da Vinci Code
4           10005  05-01-2024  Beauty Products  Neutrogena Skincare Set

   Units Sold  Unit Price  Total Revenue         Region Payment Method
0           2      999.99        1999.98  North America    Credit Card
1           1      499.99         499.99         Europe         PayPal
2           3       69.99         209.97           Asia     Debit Card
3           4       15.99          63.96  North America    Credit Card
4           1       89.99          89.99         Europe         PayPal
Saved file: Online_Sales_Analysis_MySQL.sql
```