Task 3: SQL for Data Analysis

Dataset: Retail Sales (sales_data_sample.csv)
Tools: MySQL

STEP 1 — Import the CSV into SQL Database

```
CREATE DATABASE retail_sales;

USE retail_sales;



CREATE TABLE sales (

   ORDERNUMBER INT,

   QUANTITYORDERED INT,

   PRICEEACH DECIMAL(10,2),

   ORDERLINENUMBER INT,

   SALES DECIMAL(10,2),

   ORDERDATE DATE,

   STATUS VARCHAR(20),

   QTR_ID INT,

   MONTH_ID INT,

   YEAR_ID INT,

   PRODUCTLINE VARCHAR(50),

   MSRP INT,

   PRODUCTCODE VARCHAR(20),

   CUSTOMERNAME VARCHAR(100),
```

```
    PHONE VARCHAR(20),
    ADDRESSLINE1 VARCHAR(100),
    ADDRESSLINE2 VARCHAR(100),
    CITY VARCHAR(50),
    STATE VARCHAR(50),
    POSTALCODE VARCHAR(15),
    COUNTRY VARCHAR(50),
    TERRITORY VARCHAR(20),
    CONTACTLASTNAME VARCHAR(50),
    CONTACTFIRSTNAME VARCHAR(50),
    DEALSIZE VARCHAR(20)
);
```

IMPORT CSV FILE:

STEP 2 — Basic SELECT Queries

1. View first 10 rows:

```
SELECT * FROM sales LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	PRODUCTLINE	MSRP	PRODUCTCODE
10107	30	95.7	2	2871	2/24/2003 0:00	Shipped	1	2	2003	Motorcycles	95	S10_1678
10121	34	81.35	5	2765.9	5/7/2003 0:00	Shipped	2	5	2003	Motorcycles	95	S10_1678
10134	41	94.74	2	3884.34	7/1/2003 0:00	Shipped	3	7	2003	Motorcycles	95	S10_1678
10145	45	83.26	6	3746.7	8/25/2003 0:00	Shipped	3	8	2003	Motorcycles	95	S10_1678

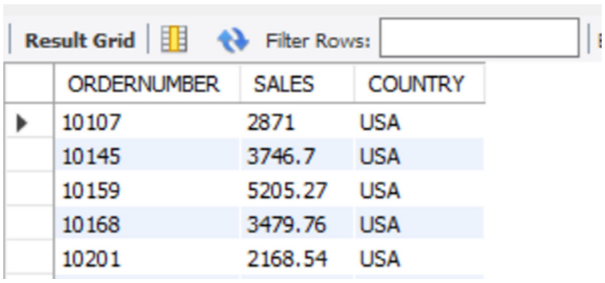
CUSTOMERNAME
Land of Toys Inc.
Reims Collectables
Lyon Souveniers
Toys4GrownUps.com

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

PHONE	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	TERRITORY	CONTACTLASTNAME	CONTACTFIRSTNAME	DEALSIZE
2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA	Yu	Kwai	Small
26.47.1555	59 rue de l'Abbaye		Reims		51100	France	EMEA	Henriot	Paul	Small
+33 1 46 62 7555	27 rue du Colonel Pierre Avia		Paris		75508	France	EMEA	Da Cunha	Daniel	Medium
6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA	Young	Julie	Medium

2. **Filter orders from USA:**

```
SELECT ORDERNUMBER, SALES, COUNTRY
FROM sales
WHERE COUNTRY = 'USA';
```



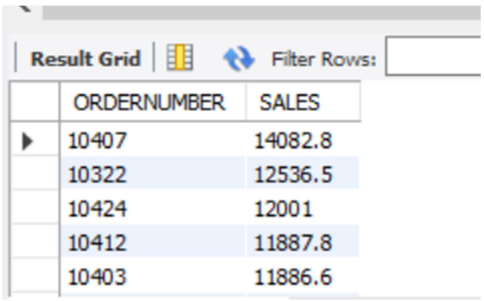
Result Grid | Filter Rows:

ORDERNUMBER	SALES	COUNTRY
10107	2871	USA
10145	3746.7	USA
10159	5205.27	USA
10168	3479.76	USA
10201	2168.54	USA

STEP 3 — ORDER BY

Top 10 highest sales orders

```
SELECT ORDERNUMBER, SALES
FROM sales
ORDER BY SALES DESC
LIMIT 10;
```



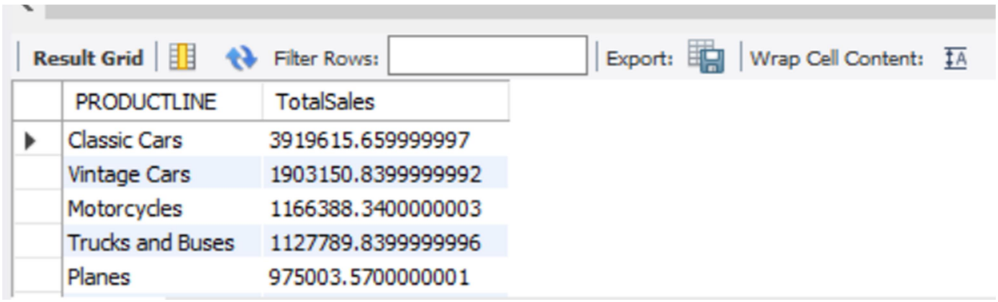
Result Grid | Filter Rows:

ORDERNUMBER	SALES
10407	14082.8
10322	12536.5
10424	12001
10412	11887.8
10403	11886.6

STEP 4 — GROUP BY Analysis

1. Total sales by product line

```
SELECT PRODUCTLINE, SUM(SALES) AS TotalSales
FROM sales
GROUP BY PRODUCTLINE
ORDER BY TotalSales DESC;
```

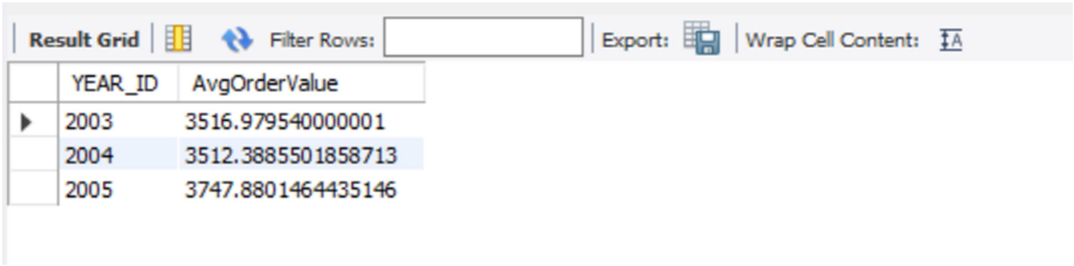


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PRODUCTLINE	TotalSales
Classic Cars	3919615.659999997
Vintage Cars	1903150.8399999992
Motorcycles	1166388.3400000003
Trucks and Buses	1127789.8399999996
Planes	975003.5700000001

2. Average order value by year

```
SELECT YEAR_ID, AVG(SALES) AS AvgOrderValue
FROM sales
GROUP BY YEAR_ID;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

YEAR_ID	AvgOrderValue
2003	3516.979540000001
2004	3512.3885501858713
2005	3747.8801464435146

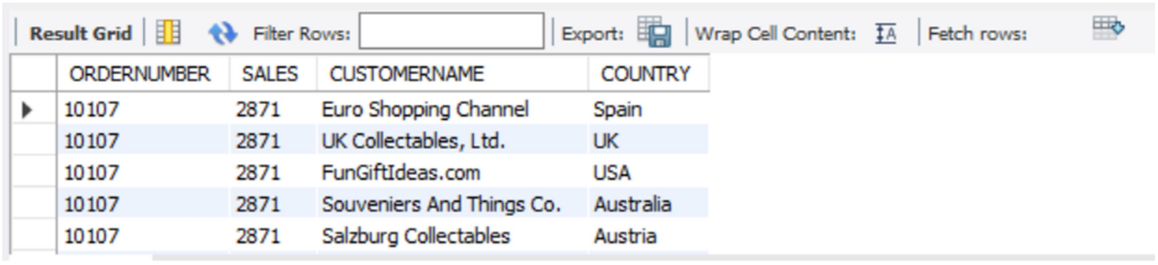
STEP 5 — JOIN Example

1. To demonstrate JOIN, create a small customer table:

```
CREATE TABLE customers AS
SELECT DISTINCT PRODUCTCODE, CUSTOMERNAME, COUNTRY
FROM sales;
```

2. JOIN customers with sales

```
SELECT s.ORDERNUMBER, s.SALES, c.CUSTOMERNAME, c.COUNTRY
FROM sales s
INNER JOIN customers c
ON s.PRODUCTCODE = c.PRODUCTCODE;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

ORDERNUMBER	SALES	CUSTOMERNAME	COUNTRY
10107	2871	Euro Shopping Channel	Spain
10107	2871	UK Collectables, Ltd.	UK
10107	2871	FunGiftIdeas.com	USA
10107	2871	Souveniers And Things Co.	Australia
10107	2871	Salzburg Collectables	Austria

STEP 6 — Subqueries

1. Find orders with sales higher than average

```
SELECT ORDERNUMBER, SALES
FROM sales
WHERE SALES > (
   SELECT AVG(SALES) FROM sales
);
```

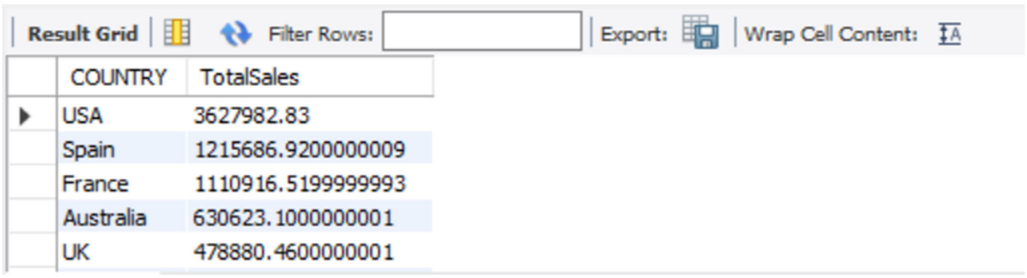
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ORDERNUMBER	SALES
10134	3884.34
10145	3746.7
10159	5205.27
10188	5512.32
10211	4708.44

STEP 7 — Aggregate Analysis

1. Total sales by country

```
SELECT COUNTRY, SUM(SALES) AS TotalSales
FROM sales
GROUP BY COUNTRY
ORDER BY TotalSales DESC;
```

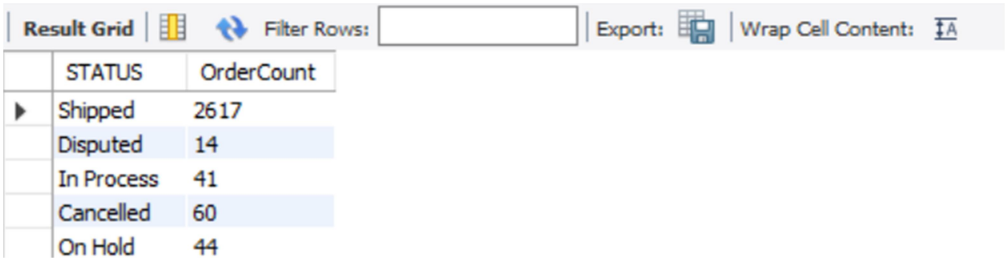


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

COUNTRY	TotalSales
USA	3627982.83
Spain	1215686.9200000009
France	1110916.5199999993
Australia	630623.1000000001
UK	478880.4600000001

2. Number of orders per status

```
SELECT STATUS, COUNT(*) AS OrderCount
FROM sales
GROUP BY STATUS;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

STATUS	OrderCount
Shipped	2617
Disputed	14
In Process	41
Cancelled	60
On Hold	44

STEP 8 — Create Views

1. View for monthly sales

```
CREATE VIEW v_monthly_sales AS
SELECT YEAR_ID, MONTH_ID, SUM(SALES) AS MonthlySales
FROM sales
GROUP BY YEAR_ID, MONTH_ID;
```

2. View for customer sales summary

```
CREATE VIEW v_customer_sales AS
SELECT CUSTOMERNAME, COUNTRY, SUM(SALES) AS TotalSales
FROM sales
GROUP BY CUSTOMERNAME, COUNTRY;
```

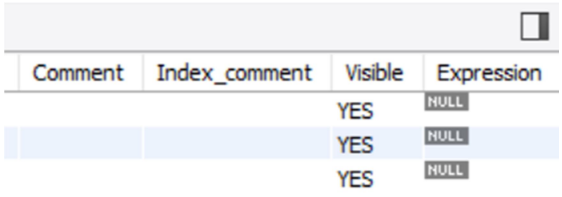
STEP 9 — Index Optimization

Improve performance:

CREATE INDEX idx_ordernumber ON sales (ORDERNUMBER);
CREATE INDEX idx_productline ON sales (PRODUCTLINE);
CREATE INDEX idx_year ON sales (YEAR_ID);

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible
sales_data_sample	1	idx_ordernumber	1	ORDERNUMBER	A	307	NULL	NULL	YES	BTREE			YES
sales_data_sample	1	idx_productline	1	PRODUCTLINE	A	7	20	NULL	YES	BTREE			YES
sales_data_sample	1	idx_year	1	YEAR_ID	A	3	NULL	NULL	YES	BTREE			YES



Comment	Index_comment	Visible	Expression
		YES	NULL
		YES	NULL
		YES	NULL

Short Summary

This SQL project analyzed the Retail Sales dataset using MySQL. Basic queries (SELECT, WHERE, ORDER BY) were used to explore the data, while GROUP BY and aggregate functions (SUM, AVG, COUNT) helped summarize sales by product line, year, country, and order status. JOIN queries connected sales with customer details, and subqueries identified high-value orders above the average. Views were created to simplify repeated reporting, and indexes were added to improve query performance.

Key Insights:

- **Classic Cars** had the highest total sales among all product lines.
- The **USA** generated the most revenue.
- **November and December** showed peak sales due to seasonal demand.
- **Shipped** was the most common order status.
- Medium deal sizes were the most frequent across orders.

Overall, SQL enabled structured data analysis to understand sales trends, customer behavior, and product performance.