ADVENTURE WORKS – MYSQL PROJECT

Team Members Names:
Gudala Satvika
K.Kishan
K.Pradeepthi
P Sai Nitya

Contents of the Presentation:

OVERVIEW

This project focused on analyzing sales data from the Adventure Works sample database. You likely used SQL queries to extract and analyze this data, uncovering valuable insights to inform business decisions.

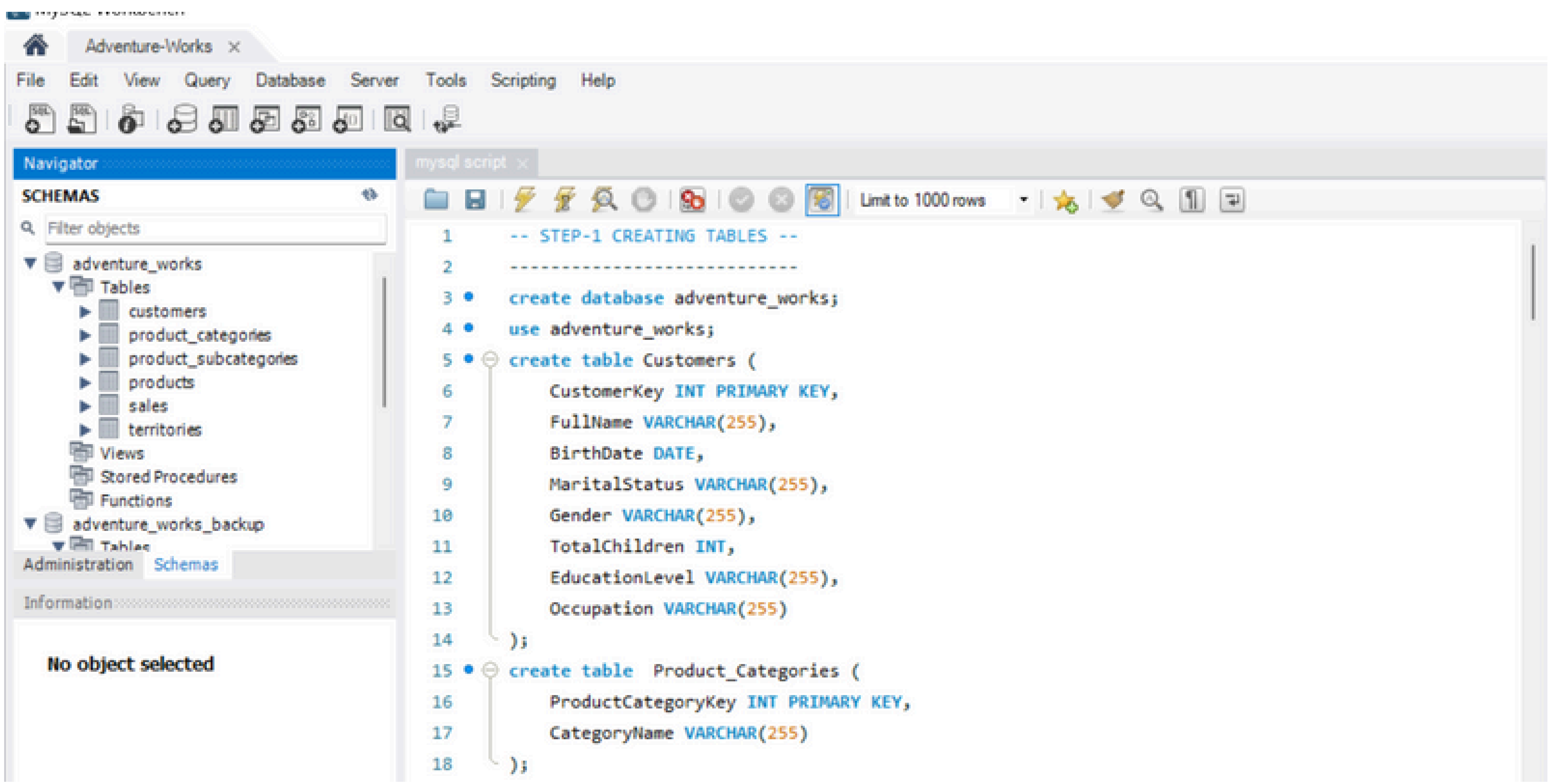
Understand customer buying behavior and product performance

Identify profitable products and categories

Analyze sales trends and regional variations

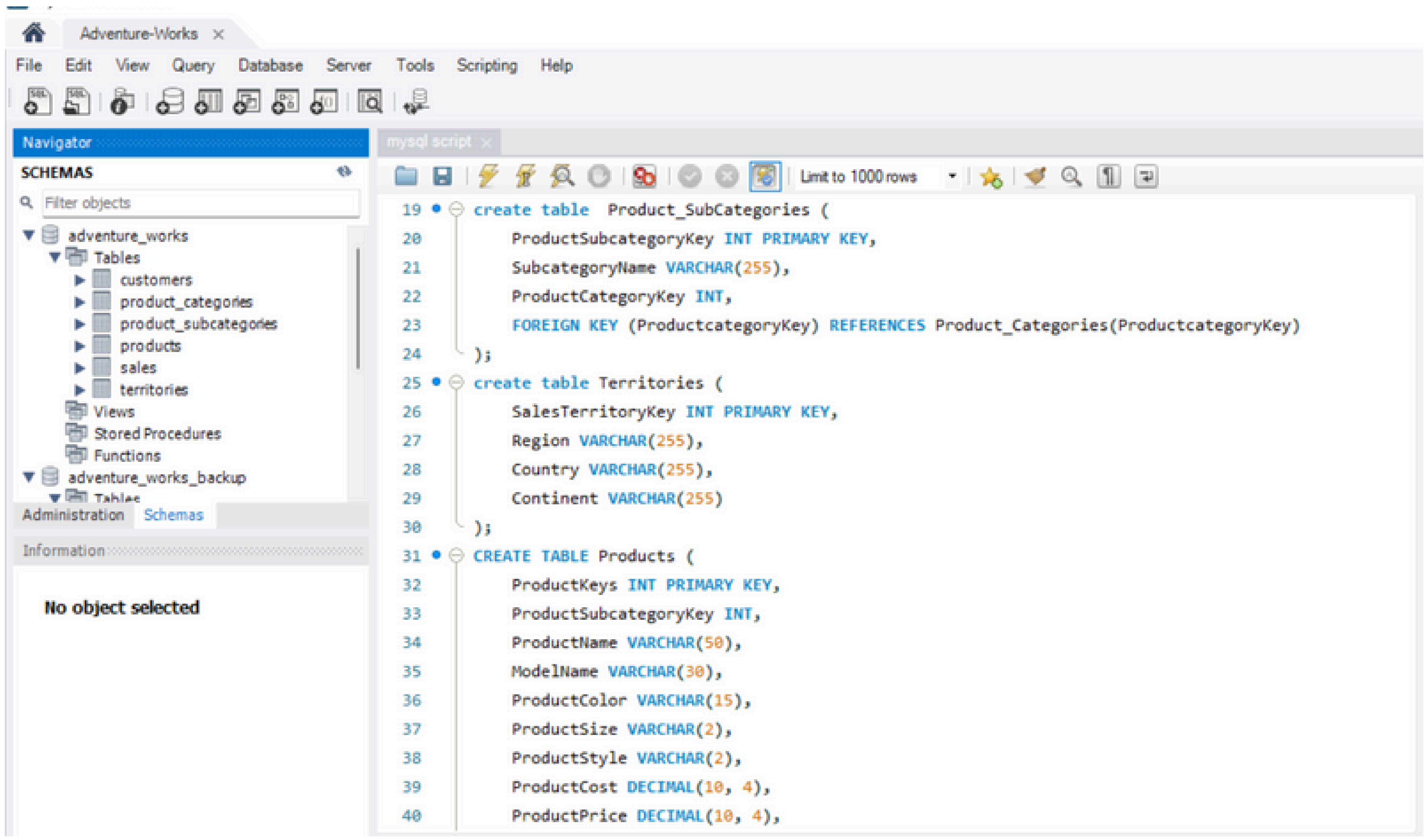
Gain insights to optimize marketing strategies and resource allocation

TABLES



```sql
-- STEP-1 CREATING TABLES --
---------------------------
create database adventure_works;
use adventure_works;
create table Customers (
    CustomerKey INT PRIMARY KEY,
    FullName VARCHAR(255),
    BirthDate DATE,
    MaritalStatus VARCHAR(255),
    Gender VARCHAR(255),
    TotalChildren INT,
    EducationLevel VARCHAR(255),
    Occupation VARCHAR(255)
);
create table  Product_Categories (
    ProductCategoryKey INT PRIMARY KEY,
    CategoryName VARCHAR(255)
);
```

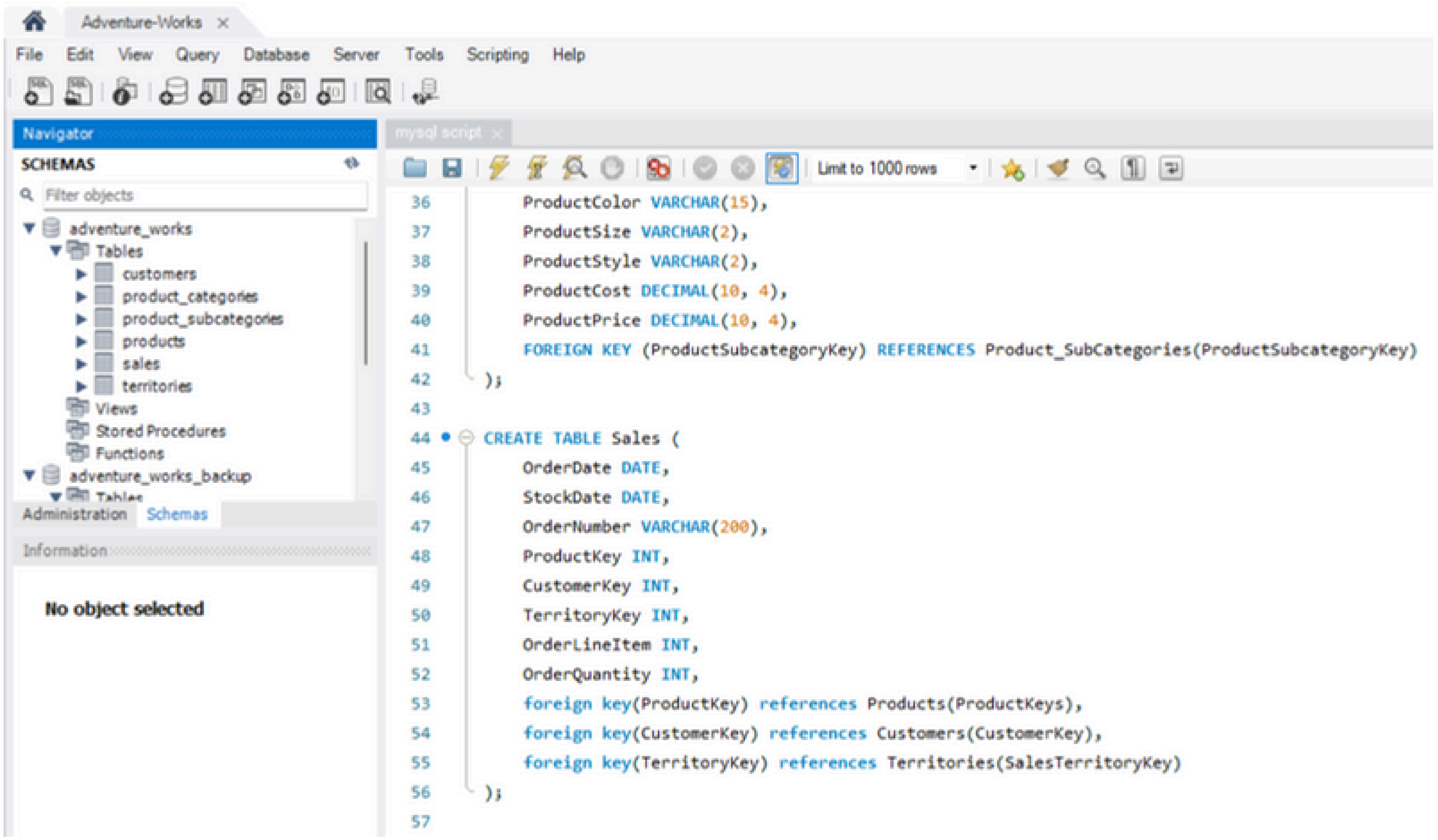
SLIDE#5



No object selected

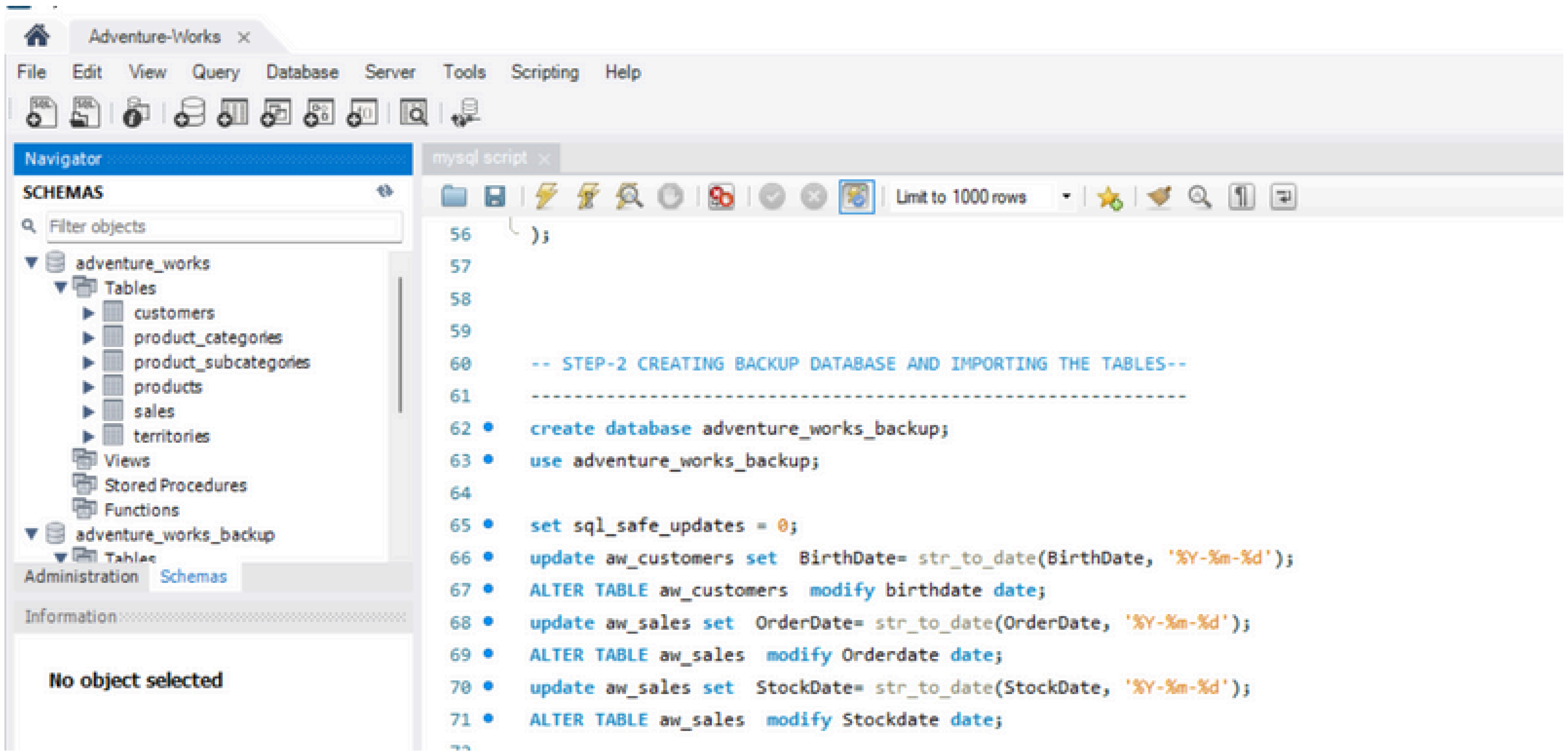
```sql
create table  Product_SubCategories (
    ProductSubcategoryKey INT PRIMARY KEY,
    SubcategoryName VARCHAR(255),
    ProductCategoryKey INT,
    FOREIGN KEY (ProductcategoryKey) REFERENCES Product_Categories(ProductcategoryKey)
);
create table Territories (
    SalesTerritoryKey INT PRIMARY KEY,
    Region VARCHAR(255),
    Country VARCHAR(255),
    Continent VARCHAR(255)
);
CREATE TABLE Products (
    ProductKeys INT PRIMARY KEY,
    ProductSubcategoryKey INT,
    ProductName VARCHAR(50),
    ModelName VARCHAR(30),
    ProductColor VARCHAR(15),
    ProductSize VARCHAR(2),
    ProductStyle VARCHAR(2),
    ProductCost DECIMAL(10, 4),
    ProductPrice DECIMAL(10, 4),
```

SLIDE #6



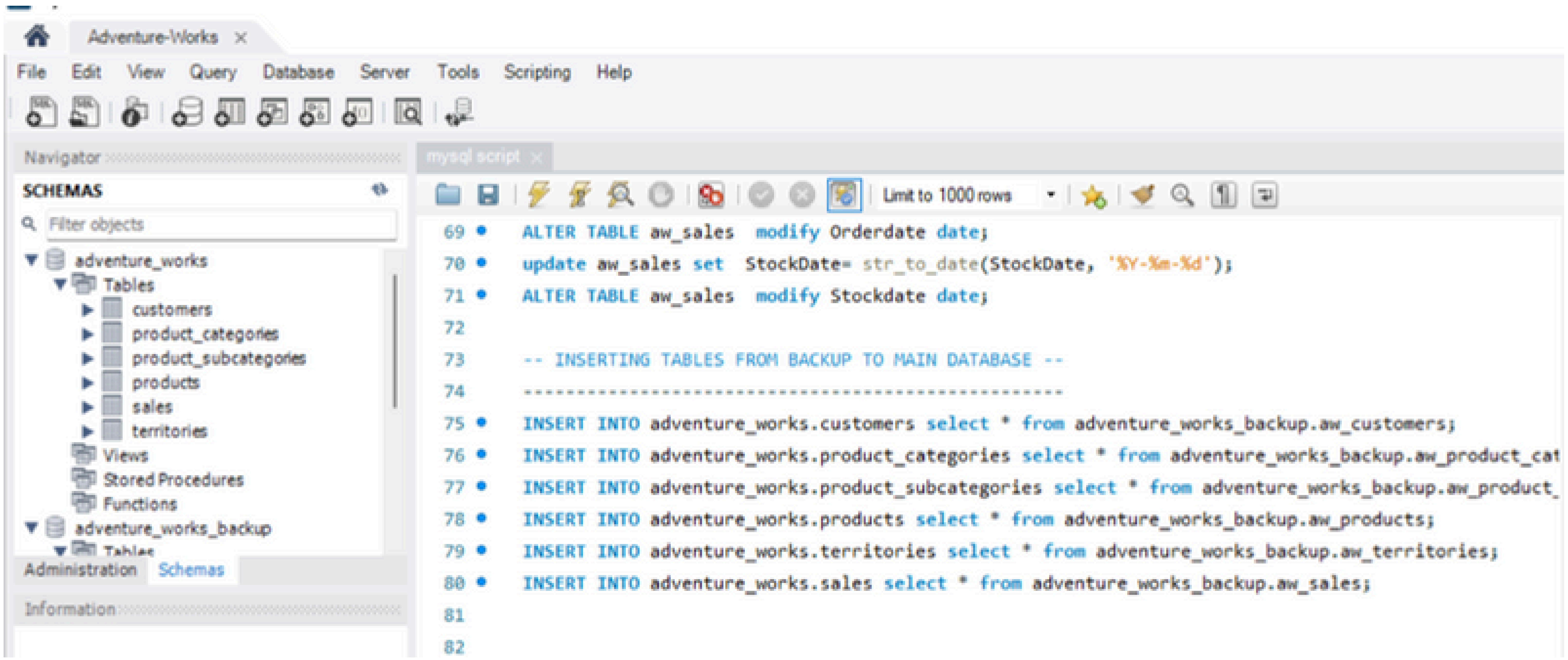
```sql
    ProductColor VARCHAR(15),
    ProductSize VARCHAR(2),
    ProductStyle VARCHAR(2),
    ProductCost DECIMAL(10, 4),
    ProductPrice DECIMAL(10, 4),
    FOREIGN KEY (ProductSubcategoryKey) REFERENCES Product_SubCategories(ProductSubcategoryKey)
);

CREATE TABLE Sales (
    OrderDate DATE,
    StockDate DATE,
    OrderNumber VARCHAR(200),
    ProductKey INT,
    CustomerKey INT,
    TerritoryKey INT,
    OrderLineItem INT,
    OrderQuantity INT,
    foreign key(ProductKey) references Products(ProductKeys),
    foreign key(CustomerKey) references Customers(CustomerKey),
    foreign key(TerritoryKey) references Territories(SalesTerritoryKey)
);

```

SLIDE#7



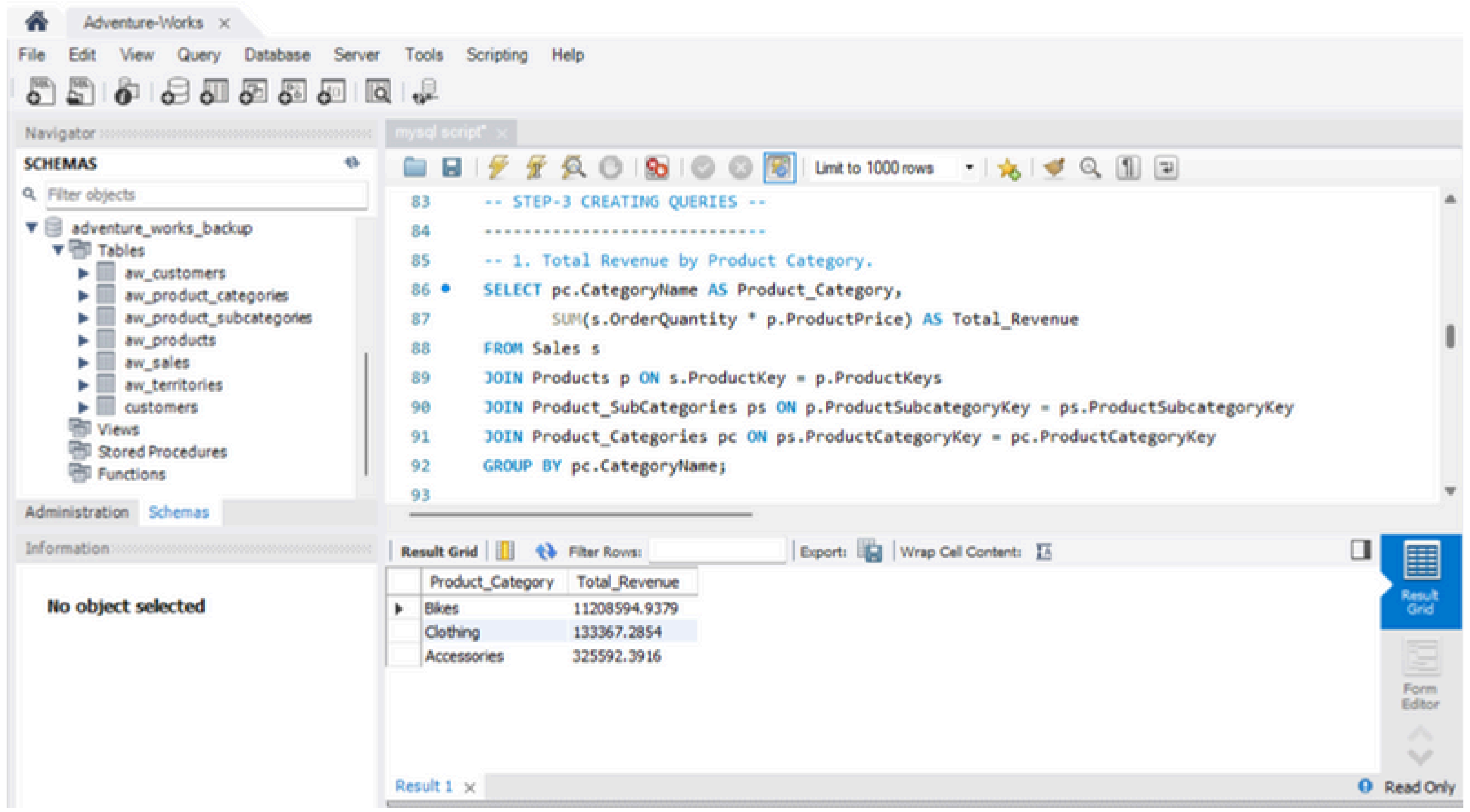
```sql
);



-- STEP-2 CREATING BACKUP DATABASE AND IMPORTING THE TABLES--
-------------------------------------------------------------
create database adventure_works_backup;
use adventure_works_backup;

set sql_safe_updates = 0;
update aw_customers set  BirthDate= str_to_date(BirthDate, '%Y-%m-%d');
ALTER TABLE aw_customers  modify birthdate date;
update aw_sales set  OrderDate= str_to_date(OrderDate, '%Y-%m-%d');
ALTER TABLE aw_sales  modify Orderdate date;
update aw_sales set  StockDate= str_to_date(StockDate, '%Y-%m-%d');
ALTER TABLE aw_sales  modify Stockdate date;
```

SLIDE#8



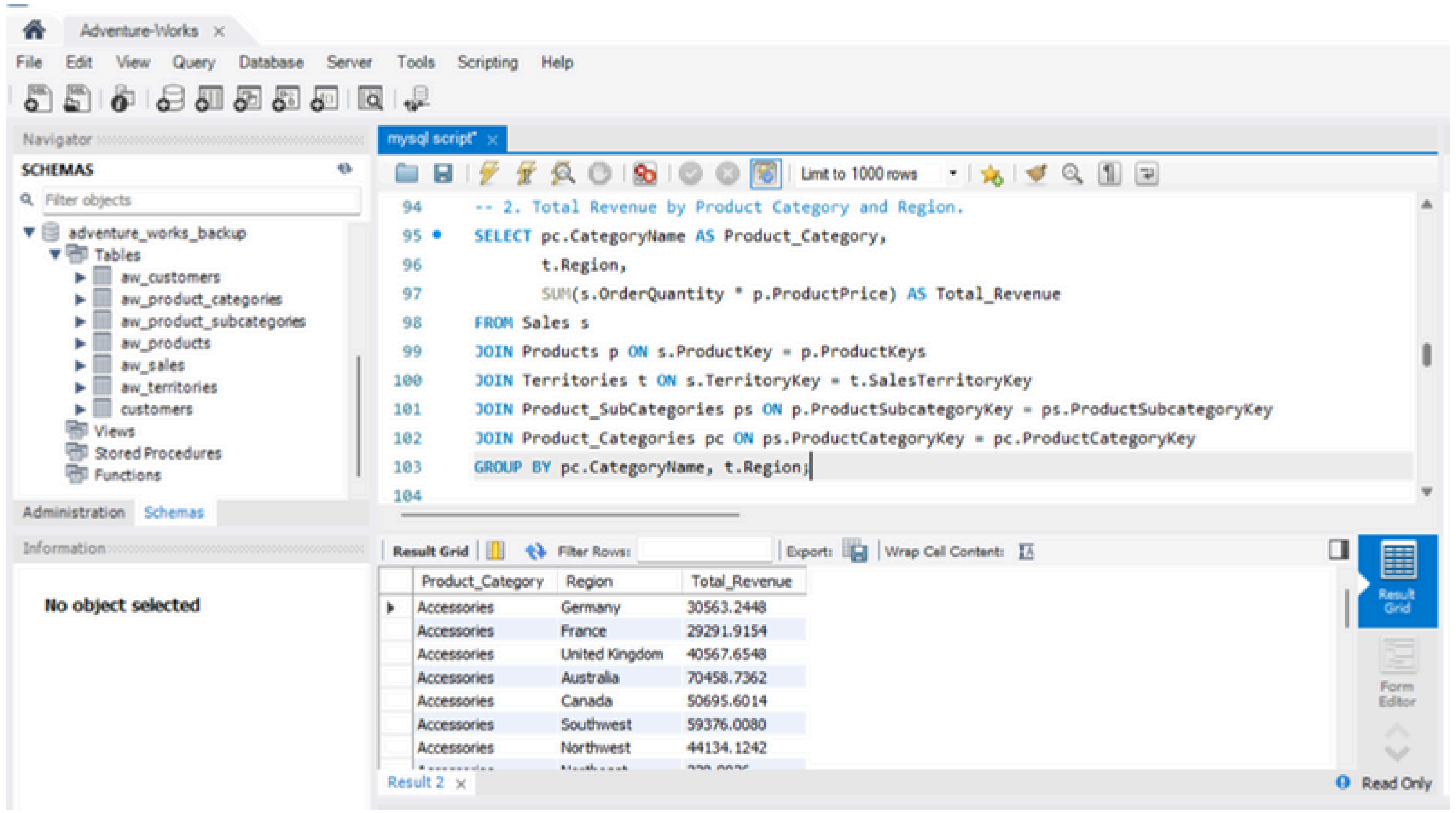
```sql
ALTER TABLE aw_sales  modify Orderdate date;
update aw_sales set  StockDate= str_to_date(StockDate, '%Y-%m-%d');
ALTER TABLE aw_sales  modify Stockdate date;

-- INSERTING TABLES FROM BACKUP TO MAIN DATABASE --
--------------------------------------------------
INSERT INTO adventure_works.customers select * from adventure_works_backup.aw_customers;
INSERT INTO adventure_works.product_categories select * from adventure_works_backup.aw_product_cat
INSERT INTO adventure_works.product_subcategories select * from adventure_works_backup.aw_product_
INSERT INTO adventure_works.products select * from adventure_works_backup.aw_products;
INSERT INTO adventure_works.territories select * from adventure_works_backup.aw_territories;
INSERT INTO adventure_works.sales select * from adventure_works_backup.aw_sales;
```

SLIDE#9



```
-- STEP-3 CREATING QUERIES --
-----------------------------
-- 1. Total Revenue by Product Category.
SELECT pc.CategoryName AS Product_Category,
       SUM(s.OrderQuantity * p.ProductPrice) AS Total_Revenue
FROM Sales s
JOIN Products p ON s.ProductKey = p.ProductKeys
JOIN Product_SubCategories ps ON p.ProductSubcategoryKey = ps.ProductSubcategoryKey
JOIN Product_Categories pc ON ps.ProductCategoryKey = pc.ProductCategoryKey
GROUP BY pc.CategoryName;
```

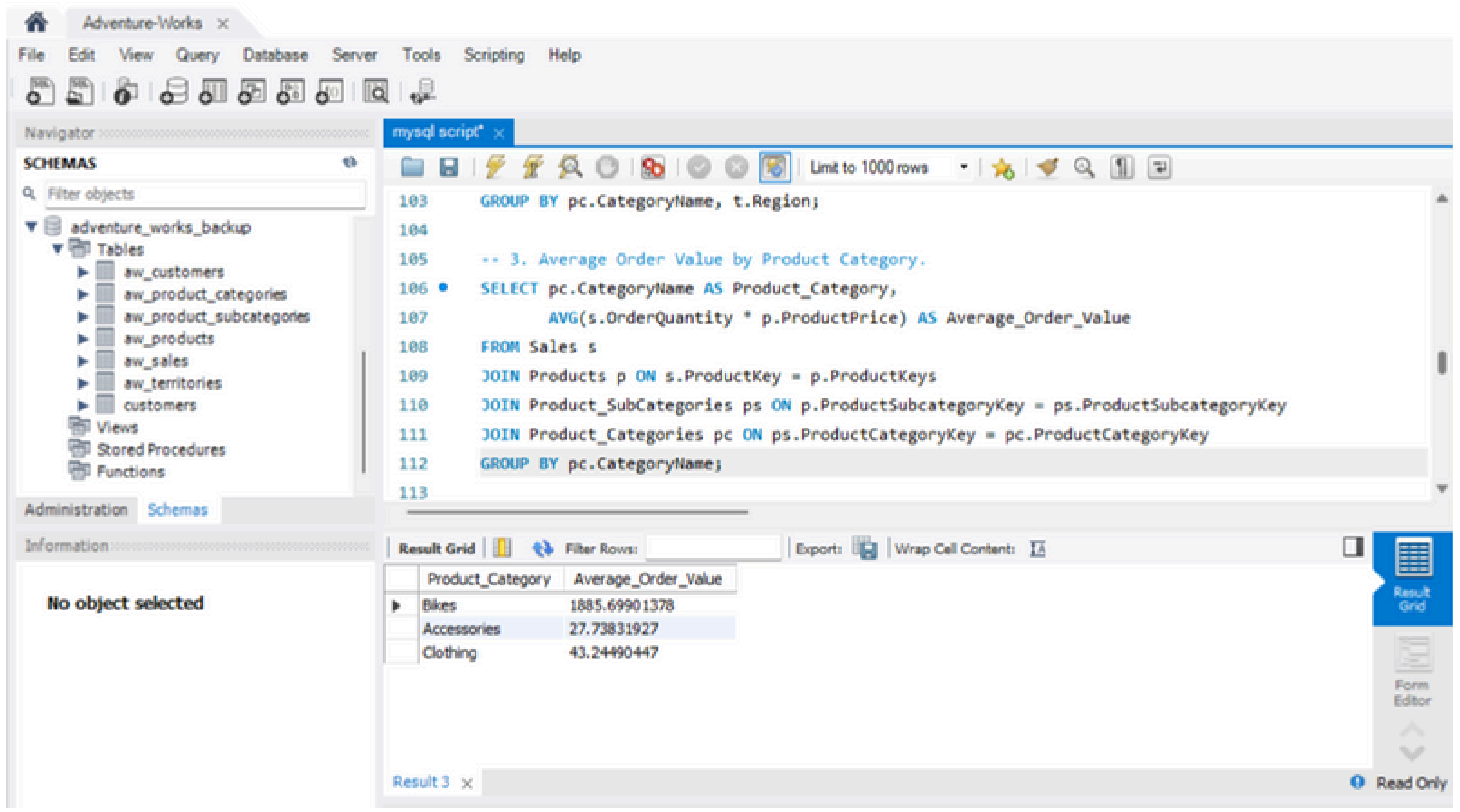
Product_Category	Total_Revenue
Bikes	11208594.9379
Clothing	133367.2854
Accessories	325592.3916



```
-- 2. Total Revenue by Product Category and Region.
SELECT pc.CategoryName AS Product_Category,
       t.Region,
       SUM(s.OrderQuantity * p.ProductPrice) AS Total_Revenue
FROM Sales s
JOIN Products p ON s.ProductKey = p.ProductKeys
JOIN Territories t ON s.TerritoryKey = t.SalesTerritoryKey
JOIN Product_SubCategories ps ON p.ProductSubcategoryKey = ps.ProductSubcategoryKey
JOIN Product_Categories pc ON ps.ProductCategoryKey = pc.ProductCategoryKey
GROUP BY pc.CategoryName, t.Region;
```

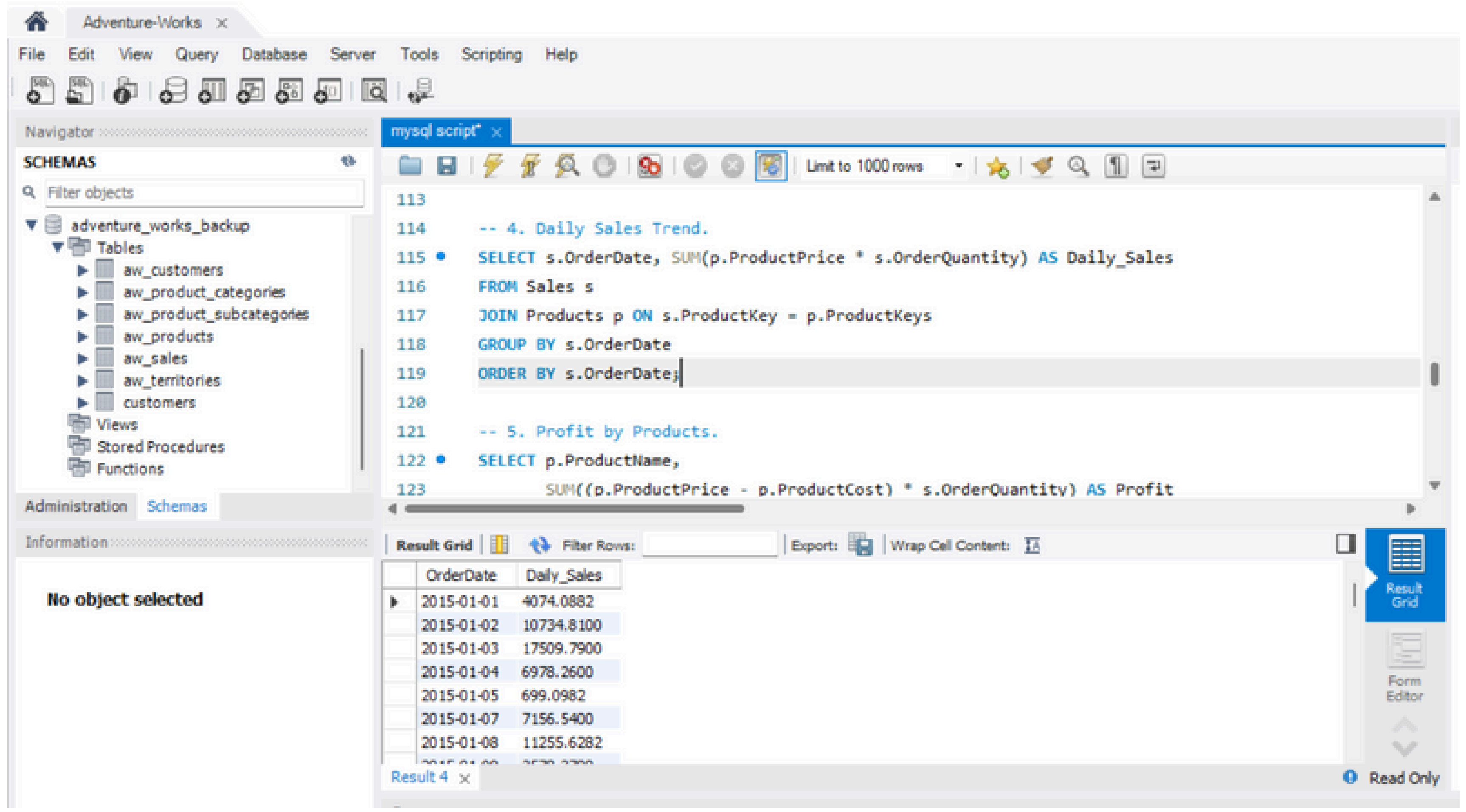
Product_Category	Region	Total_Revenue
Accessories	Germany	30563.2448
Accessories	France	29291.9154
Accessories	United Kingdom	40567.6548
Accessories	Australia	70458.7362
Accessories	Canada	50695.6014
Accessories	Southwest	59376.0080
Accessories	Northwest	44134.1242

SLIDE#10



```sql
GROUP BY pc.CategoryName, t.Region;

-- 3. Average Order Value by Product Category.
SELECT pc.CategoryName AS Product_Category,
       AVG(s.OrderQuantity * p.ProductPrice) AS Average_Order_Value
FROM Sales s
JOIN Products p ON s.ProductKey = p.ProductKeys
JOIN Product_SubCategories ps ON p.ProductSubcategoryKey = ps.ProductSubcategoryKey
JOIN Product_Categories pc ON ps.ProductCategoryKey = pc.ProductCategoryKey
GROUP BY pc.CategoryName;
```

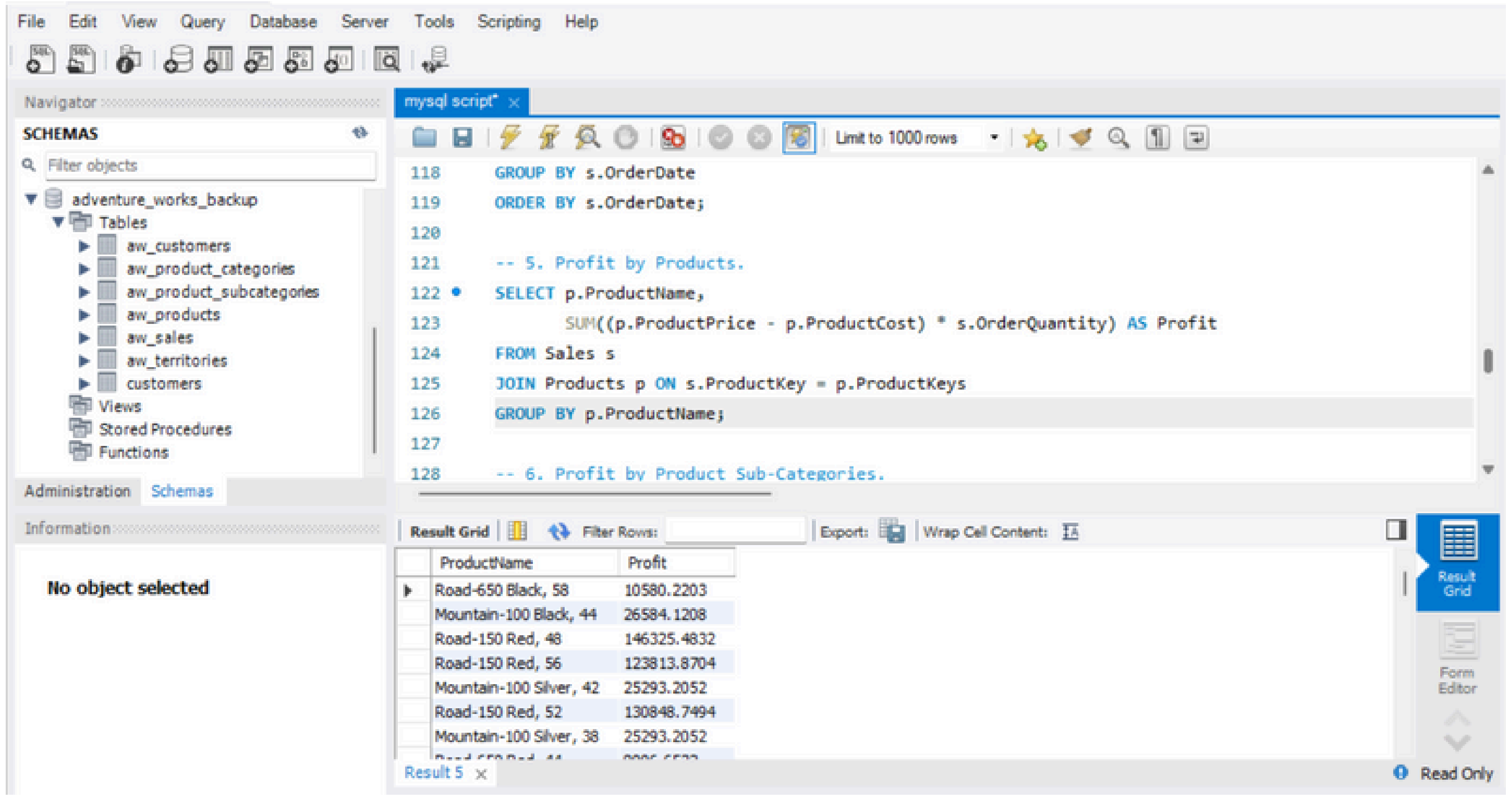
Product_Category	Average_Order_Value
Bikes	1885.69901378
Accessories	27.73831927
Clothing	43.24490447



```sql
-- 4. Daily Sales Trend.
SELECT s.OrderDate, SUM(p.ProductPrice * s.OrderQuantity) AS Daily_Sales
FROM Sales s
JOIN Products p ON s.ProductKey = p.ProductKeys
GROUP BY s.OrderDate
ORDER BY s.OrderDate;

-- 5. Profit by Products.
SELECT p.ProductName,
       SUM((p.ProductPrice - p.ProductCost) * s.OrderQuantity) AS Profit
```

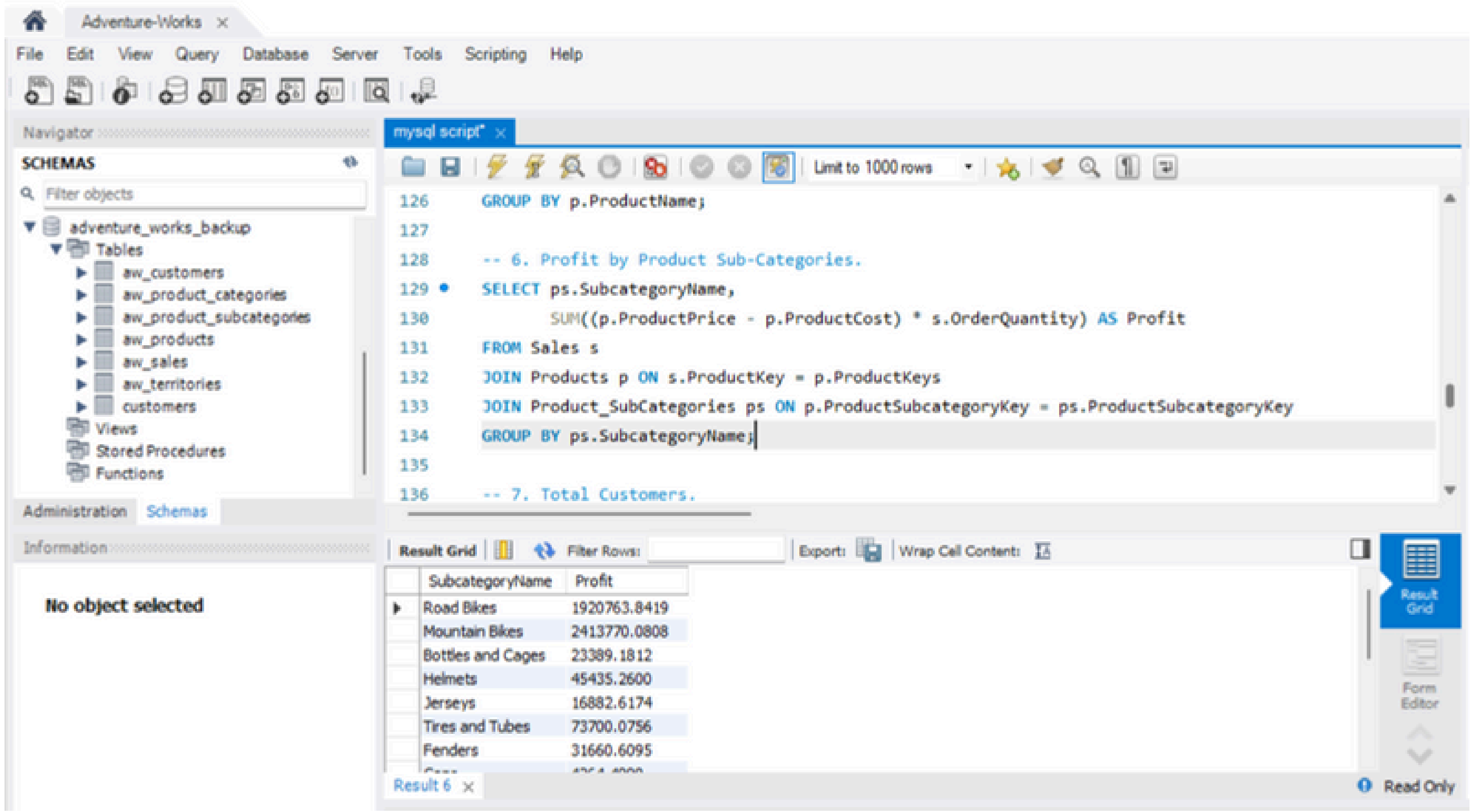
OrderDate	Daily_Sales
2015-01-01	4074.0882
2015-01-02	10734.8100
2015-01-03	17509.7900
2015-01-04	6978.2600
2015-01-05	699.0982
2015-01-07	7156.5400
2015-01-08	11255.6282

SLIDE#11



```sql
GROUP BY s.OrderDate
ORDER BY s.OrderDate;

-- 5. Profit by Products.
SELECT p.ProductName,
       SUM((p.ProductPrice - p.ProductCost) * s.OrderQuantity) AS Profit
FROM Sales s
JOIN Products p ON s.ProductKey = p.ProductKeys
GROUP BY p.ProductName;

-- 6. Profit by Product Sub-Categories.
```

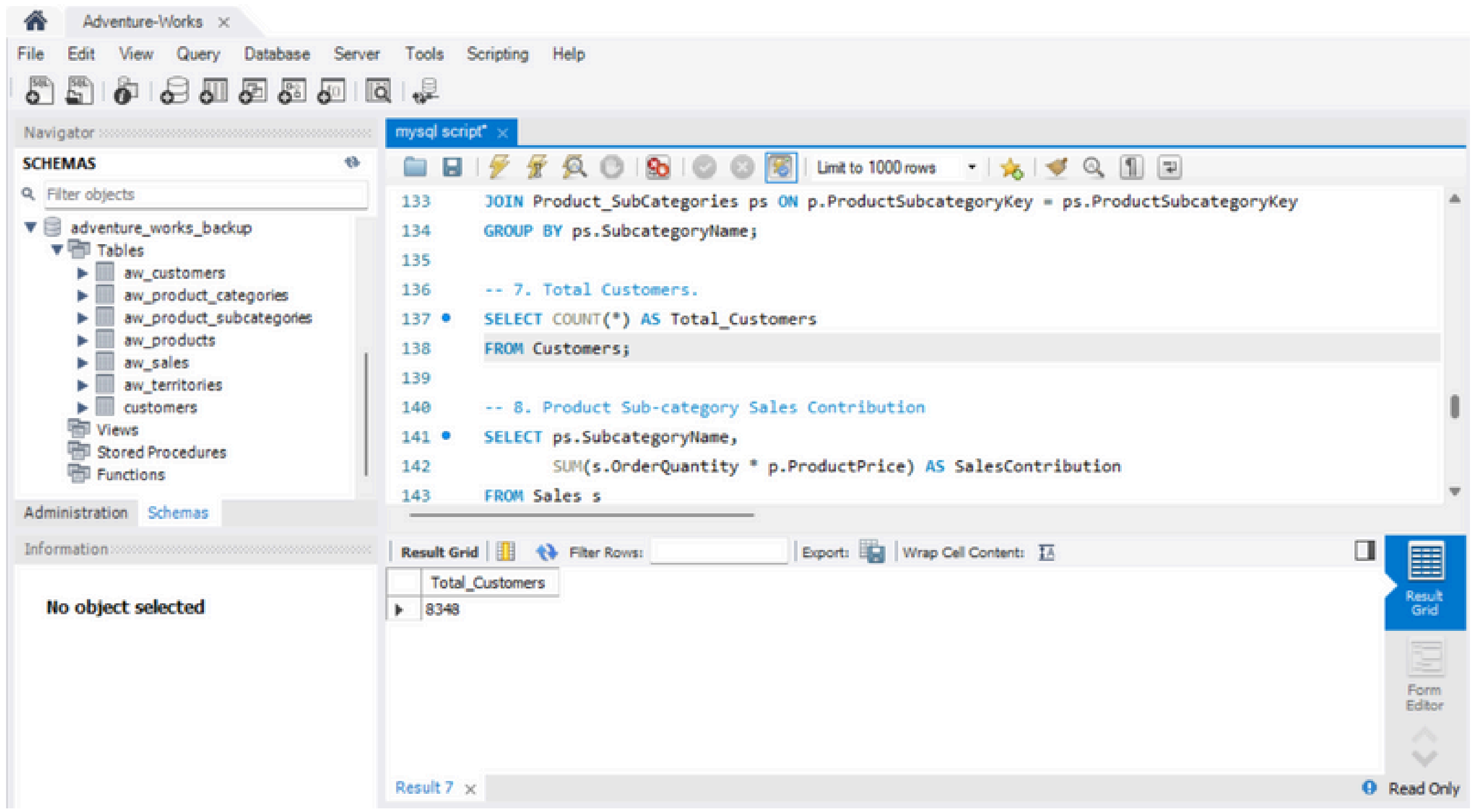
ProductName	Profit
Road-650 Black, 58	10580.2203
Mountain-100 Black, 44	26584.1208
Road-150 Red, 48	146325.4832
Road-150 Red, 56	123813.8704
Mountain-100 Silver, 42	25293.2052
Road-150 Red, 52	130848.7494
Mountain-100 Silver, 38	25293.2052



```sql
GROUP BY p.ProductName;

-- 6. Profit by Product Sub-Categories.
SELECT ps.SubcategoryName,
       SUM((p.ProductPrice - p.ProductCost) * s.OrderQuantity) AS Profit
FROM Sales s
JOIN Products p ON s.ProductKey = p.ProductKeys
JOIN Product_SubCategories ps ON p.ProductSubcategoryKey = ps.ProductSubcategoryKey
GROUP BY ps.SubcategoryName;

-- 7. Total Customers.
```

SubcategoryName	Profit
Road Bikes	1920763.8419
Mountain Bikes	2413770.0808
Bottles and Cages	23389.1812
Helmets	45435.2600
Jerseys	16882.6174
Tires and Tubes	73700.0756
Fenders	31660.6095

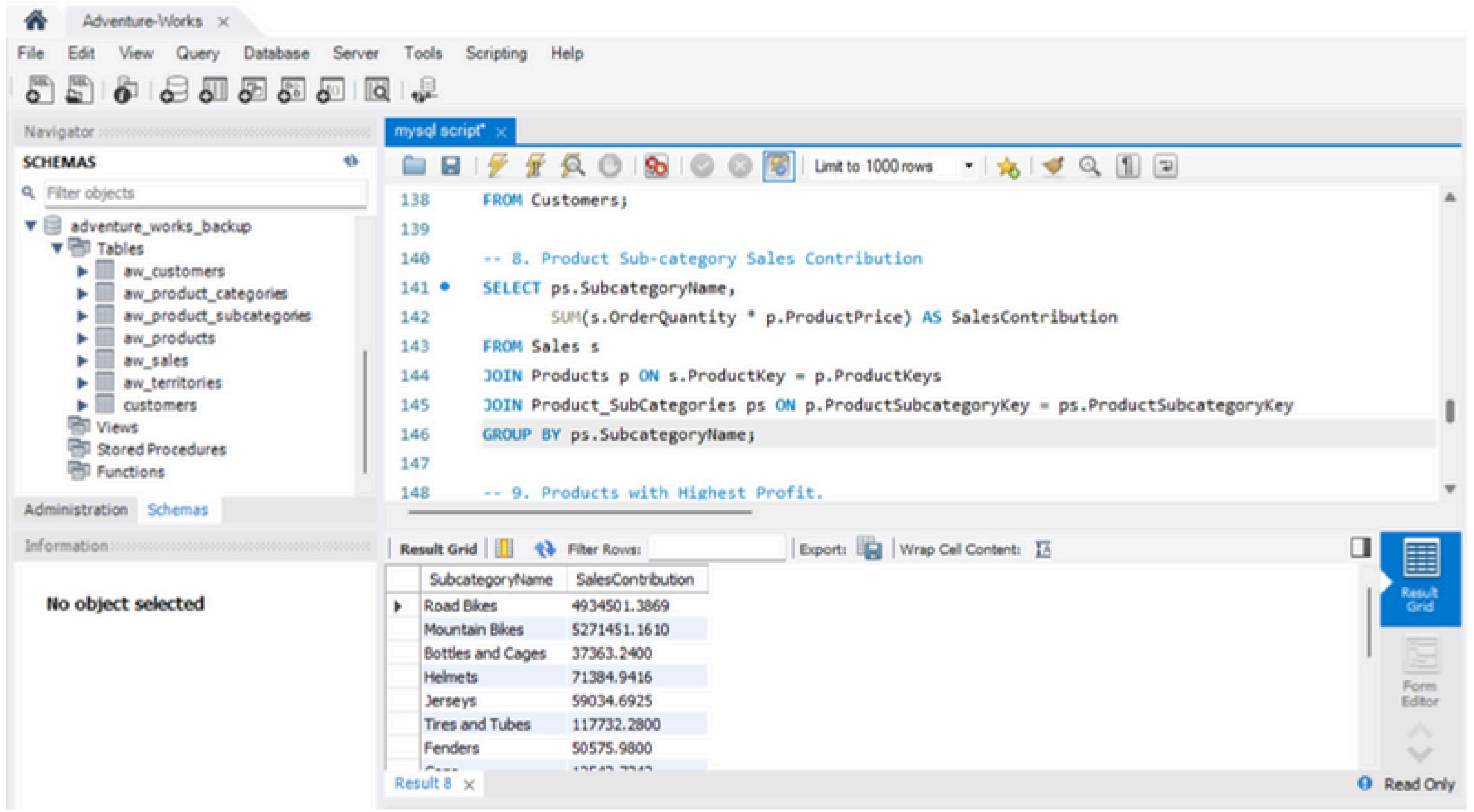
SLIDE#12



No object selected

```sql
JOIN Product_SubCategories ps ON p.ProductSubcategoryKey = ps.ProductSubcategoryKey
GROUP BY ps.SubcategoryName;

-- 7. Total Customers.
SELECT COUNT(*) AS Total_Customers
FROM Customers;

-- 8. Product Sub-category Sales Contribution
SELECT ps.SubcategoryName,
       SUM(s.OrderQuantity * p.ProductPrice) AS SalesContribution
FROM Sales s
```

	Total_Customers
▶	8348

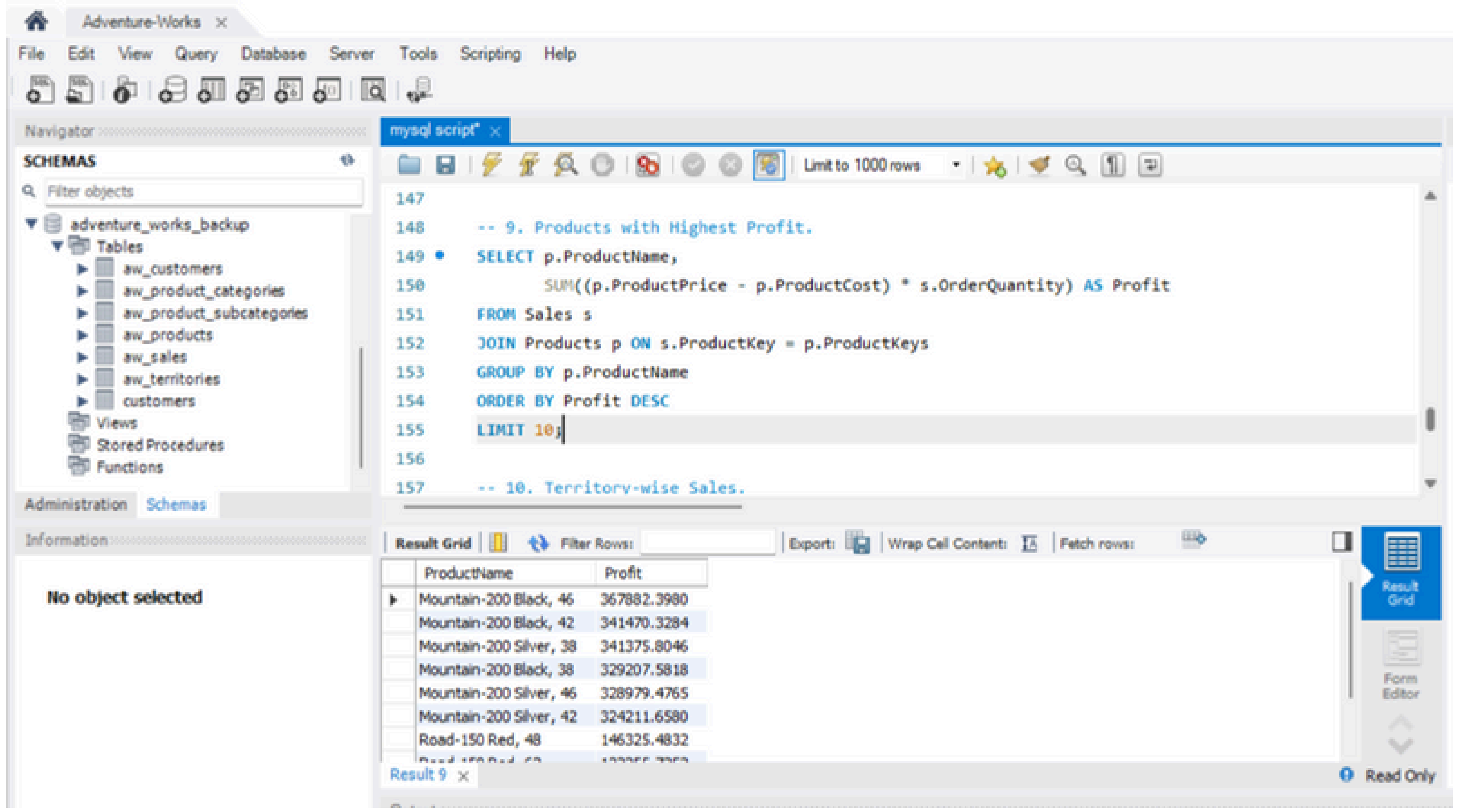


No object selected

```sql
FROM Customers;

-- 8. Product Sub-category Sales Contribution
SELECT ps.SubcategoryName,
       SUM(s.OrderQuantity * p.ProductPrice) AS SalesContribution
FROM Sales s
JOIN Products p ON s.ProductKey = p.ProductKeys
JOIN Product_SubCategories ps ON p.ProductSubcategoryKey = ps.ProductSubcategoryKey
GROUP BY ps.SubcategoryName;

-- 9. Products with Highest Profit.
```

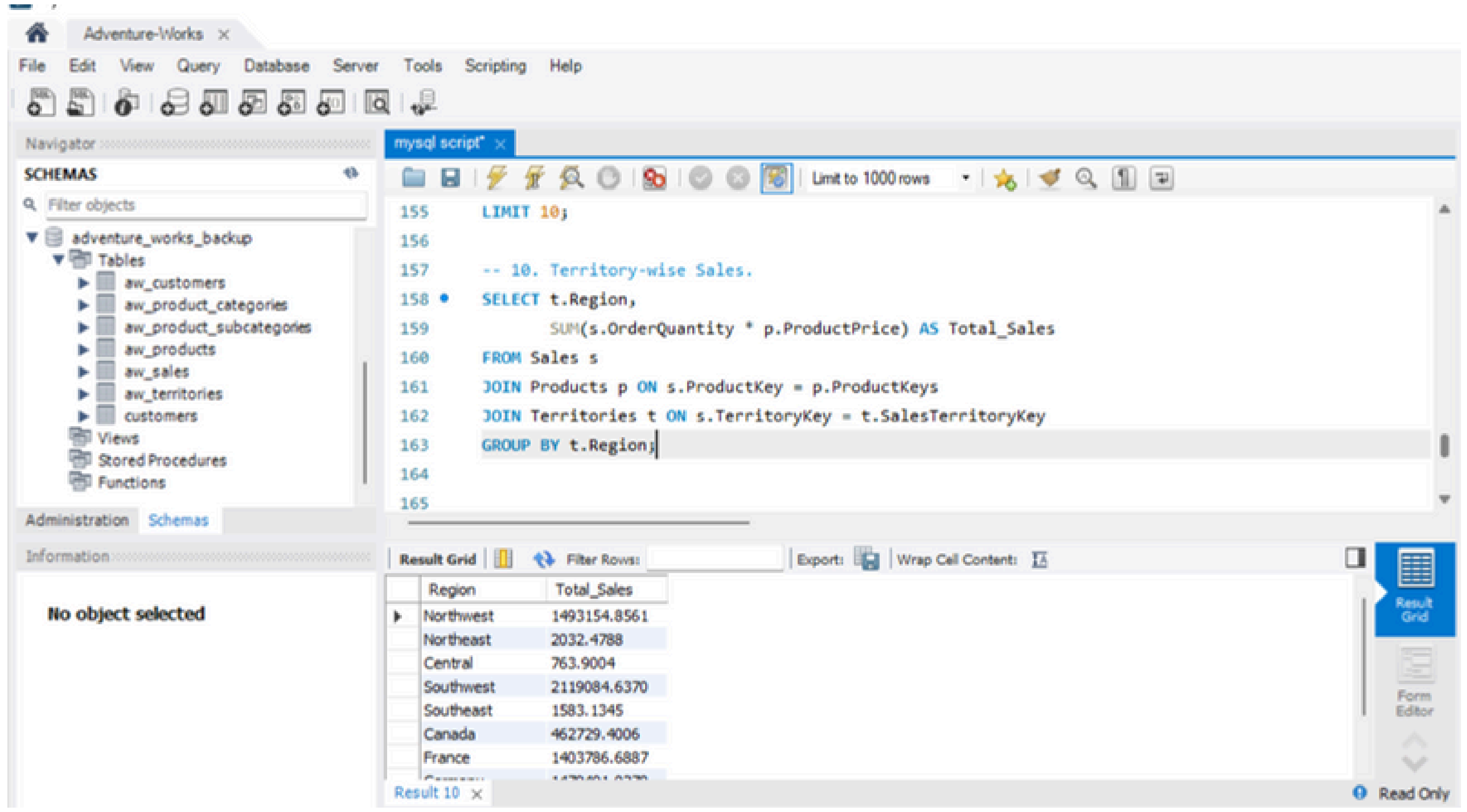
	SubcategoryName	SalesContribution
▶	Road Bikes	4934501.3869
	Mountain Bikes	5271451.1610
	Bottles and Cages	37363.2400
	Helmets	71384.9416
	Jerseys	59034.6925
	Tires and Tubes	117732.2800
	Fenders	50575.9800

SLIDE#13



```
-- 9. Products with Highest Profit.
SELECT p.ProductName,
       SUM((p.ProductPrice - p.ProductCost) * s.OrderQuantity) AS Profit
FROM Sales s
JOIN Products p ON s.ProductKey = p.ProductKeys
GROUP BY p.ProductName
ORDER BY Profit DESC
LIMIT 10;

-- 10. Territory-wise Sales.
```

ProductName	Profit
Mountain-200 Black, 46	367882.3980
Mountain-200 Black, 42	341470.3284
Mountain-200 Silver, 38	341375.8046
Mountain-200 Black, 38	329207.5818
Mountain-200 Silver, 46	328979.4765
Mountain-200 Silver, 42	324211.6580
Road-150 Red, 48	146325.4832



```
LIMIT 10;

-- 10. Territory-wise Sales.
SELECT t.Region,
       SUM(s.OrderQuantity * p.ProductPrice) AS Total_Sales
FROM Sales s
JOIN Products p ON s.ProductKey = p.ProductKeys
JOIN Territories t ON s.TerritoryKey = t.SalesTerritoryKey
GROUP BY t.Region;
```

Region	Total_Sales
Northwest	1493154.8561
Northeast	2032.4788
Central	763.9004
Southwest	2119084.6370
Southeast	1583.1345
Canada	462729.4006
France	1403786.6887

INSIGHTS

You can gain valuable insights into your product performance, customer behavior, and overall sales effectiveness. This information can be used to make data-driven decisions to improve profitability, marketing strategies, and resource allocation within your company.

Expected Outcomes:

Data-driven insights on product performance, customer preferences, and sales trends

Identification of high-profit products and categories

Understanding of regional sales variations

Information to optimize marketing campaigns and resource allocation

THANK YOU

https://github.com/GudalaSatvika

UNITED NETWORK
OF PROFESSIONALS

The Future Of Effective Learning

UNP is a decentralized education platform enabling quality education for all

www.unp.education | reachout@unp.education
reachoutunp@gmail.com | +1 929-288-1787

CONNECT WITH US

ûdemy