**Microsoft Applied Skills: Implement a lakehouse in Microsoft Fabric**

You need to create delta tables from the CSV files in C:\files, perform various transformation steps to shape the data, and query the data by using the SQL analytics endpoint. The solution must meet the following requirements:

* Upload the files from C:\files\sales\ to the Files folder of the Sales lakehouse, and then create a delta table for each file in the Tables folder of the Sales lakehouse. Table names must have a prefix **raw\_**.

df=spark.read.load('Files/customers.csv', format='csv', header=True)

df.write.format("delta").saveAsTable("raw\_customers")

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

df=spark.read.load('Files/customers2.csv', format='csv', header=True)

df.write.format("delta").saveAsTable("raw\_customers2")

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

df=spark.read.load('Files/orders.csv', format='csv', header=True)

df.write.format("delta").saveAsTable("raw\_orders")

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

df=spark.read.load('Files/products.csv', format='csv', header=True)

df.write.format("delta").saveAsTable("raw\_products")

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Upload C:\files\date\date.csv to the Files folder of the Sales lakehouse, and then create a delta table for a file named **date**.
* Combine the data from **raw\_customers** and **raw\_customers2** into a single table named **prod\_customers** and ensure that there are no duplicate records.

from delta.tables import DeltaTable

from pyspark.sql import functions as F

raw\_customers = DeltaTable.forName(spark, "raw\_customers")

raw\_customers2 = DeltaTable.forName(spark, "raw\_customers2")

df\_raw\_customers = raw\_customers.toDF()

df\_raw\_customers2 = raw\_customers2.toDF()

combined\_df = df\_raw\_customers.unionByName(df\_raw\_customers2)

final\_df = combined\_df.dropDuplicates()

from pyspark.sql.functions import col

from pyspark.sql.types import IntegerType,DoubleType,DateType

final\_df=final\_df.withColumn("CustomerID",col("CustomerID").cast(IntegerType()))

final\_df.write.format("delta").mode("overwrite").saveAsTable("prod\_customers")

* In **prod\_customers**, create a column named **FullName** that concatenates values from the **FirstName** and **LastName** columns.

%%sql

ALTER TABLE prod\_customers

ADD COLUMN FullName STRING;

UPDATE prod\_customers

SET FullName = CONCAT(FirstName, ' ', LastName);

* Create a new table named **prod\_orders** that contains that data from **raw\_orders**. In prod\_orders, replace the blank values in the **LineItemTotal** column with zeroes.

df = spark.sql("SELECT \* FROM Sales.dbo.raw\_orders LIMIT 1000")

df.show()

df.printSchema()

#raw\_orders\_df = spark.table("raw\_orders")

from pyspark.sql import functions as F

prod\_orders\_df = df.withColumn(

    "LineItemTotal",

    F.when(F.col("LineItemTotal") == '', 0)

    .otherwise(F.coalesce(F.col("LineItemTotal"), F.lit(0)))

)

prod\_orders\_df.show()

prod\_orders\_df.printSchema()

from pyspark.sql.functions import col

from pyspark.sql import functions as F

from pyspark.sql.types import IntegerType,DoubleType,DateType

prod\_orders\_df=prod\_orders\_df.withColumn("SalesOrderID",col("SalesOrderID").cast(IntegerType()))\

.withColumn("OrderDate",col("OrderDate").cast(DateType()))\

.withColumn("CustomerID",col("CustomerID").cast(IntegerType()))\

.withColumn("LineItem",col("LineItem").cast(IntegerType()))\

.withColumn("ProductID",col("ProductID").cast(IntegerType()))\

.withColumn("OrderQty",col("OrderQty").cast(IntegerType()))\

.withColumn("LineItemTotal",col("LineItemTotal").cast(DoubleType()))

prod\_orders\_df.show()

prod\_orders\_df.printSchema()

#prod\_orders\_df = prod\_orders\_df.withColumn("date\_col", F.to\_date("OrderDate", "MM-dd-yyyy"))

#prod\_orders\_df = prod\_orders\_df.withColumn("OrderDates", col("OrderDate").cast("date"))

#df = df.withColumn("date\_col", col("string\_col").cast("date"))

#prod\_orders\_df.show()

#prod\_orders\_df.printSchema()

prod\_orders\_df.write.format("delta").mode("overwrite").saveAsTable("prod\_orders")

* Create a new table named **prod\_products** that contains data from **raw\_products**.

df = spark.sql("SELECT \* FROM Sales.dbo.raw\_products LIMIT 1000")

df.show()

df.printSchema()

from pyspark.sql.functions import col

from pyspark.sql import functions as F

from pyspark.sql.types import IntegerType,DoubleType,DateType

df=df.withColumn("ProductID",col("ProductID").cast(IntegerType()))\

.withColumn("ListPrice",col("ListPrice").cast(DoubleType()))

df.show()

df.printSchema()

df.write.format("delta").mode("overwrite").saveAsTable("prod\_products")

* In tables that have a prefix of prod\_, change the data type of all the numeric and date columns.
* In the **Sales** lakehouse, create a view in the dbo schema that returns the sum of LineItemTotal per customer for the year 2022. The view must contain the following columns: FullName, Year, and TotalSales. TotalSales is the sum of LineItemTotal.

CREATE VIEW Sales2022 AS

SELECT c.FullName,

       YEAR(OrderDate) as YEAR,

       SUM(LineItemTotal) AS TOTALSALES

FROM prod\_customers c LEFT JOIN prod\_orders o

ON c.CustomerID = o.CustomerID

WHERE YEAR(OrderDate) = 2022

GROUP BY c.FullName,YEAR(OrderDate)

Set up the PoC environment  
projman@contoso.com

Ingest and transform data  
projman@contoso.com

Product sales analysis  
projman@contoso.com

Update Spark notebook  
projman@contoso.com

Status - Complete  
cto@contoso.com

**Subject: Product sales analysis**

From: [projman@contoso.com](mailto:)  
To: [admin-45511000@LODSPRODMCA.onmicrosoft.com](mailto:)

Mark as unread  
  
Mark as completed

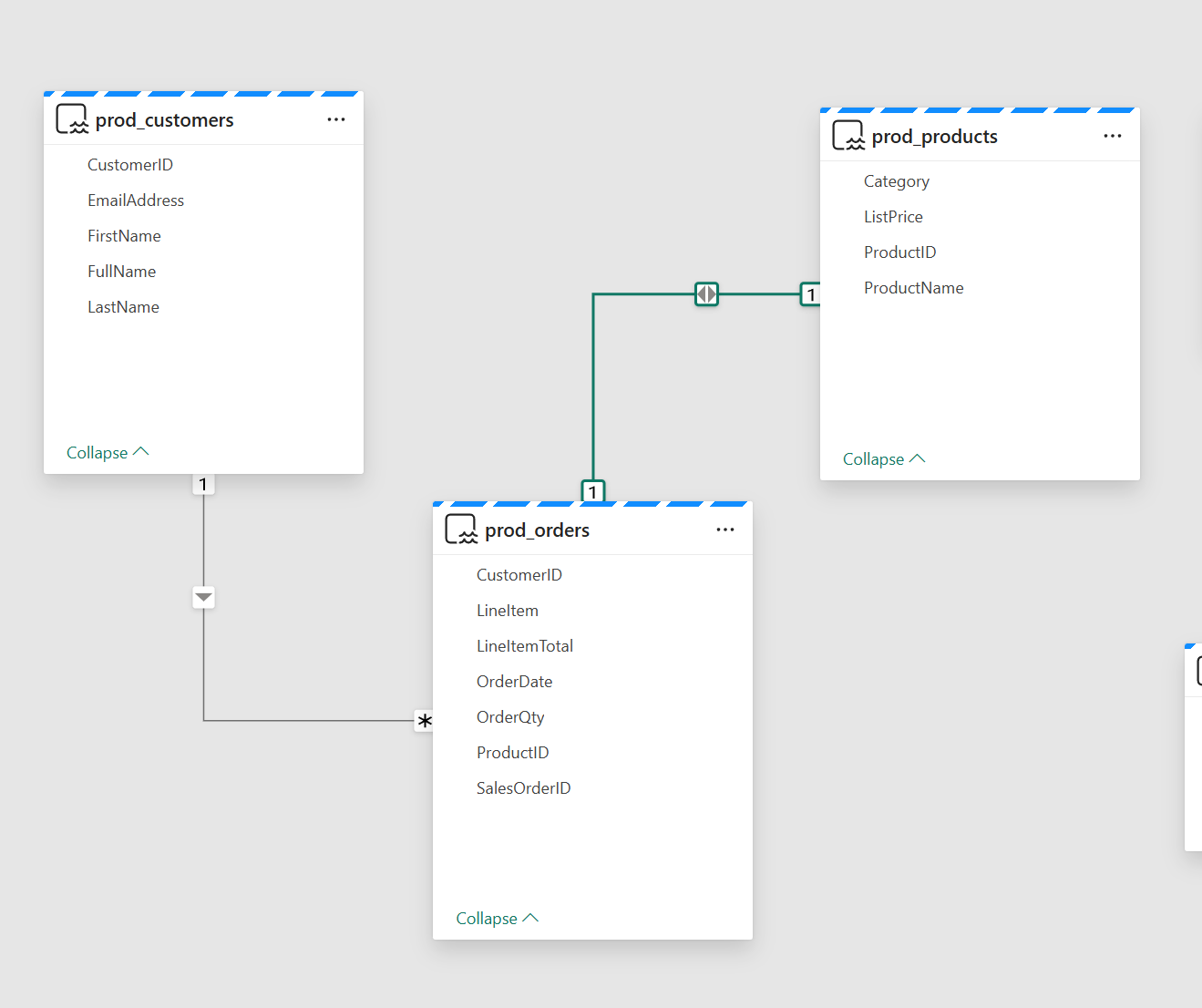
Hello,

We need to use the Sales lakehouse to analyze product sales. We need you to configure the relationships in the default semantic model, and then create a report about bike sales.

**This is your task:**

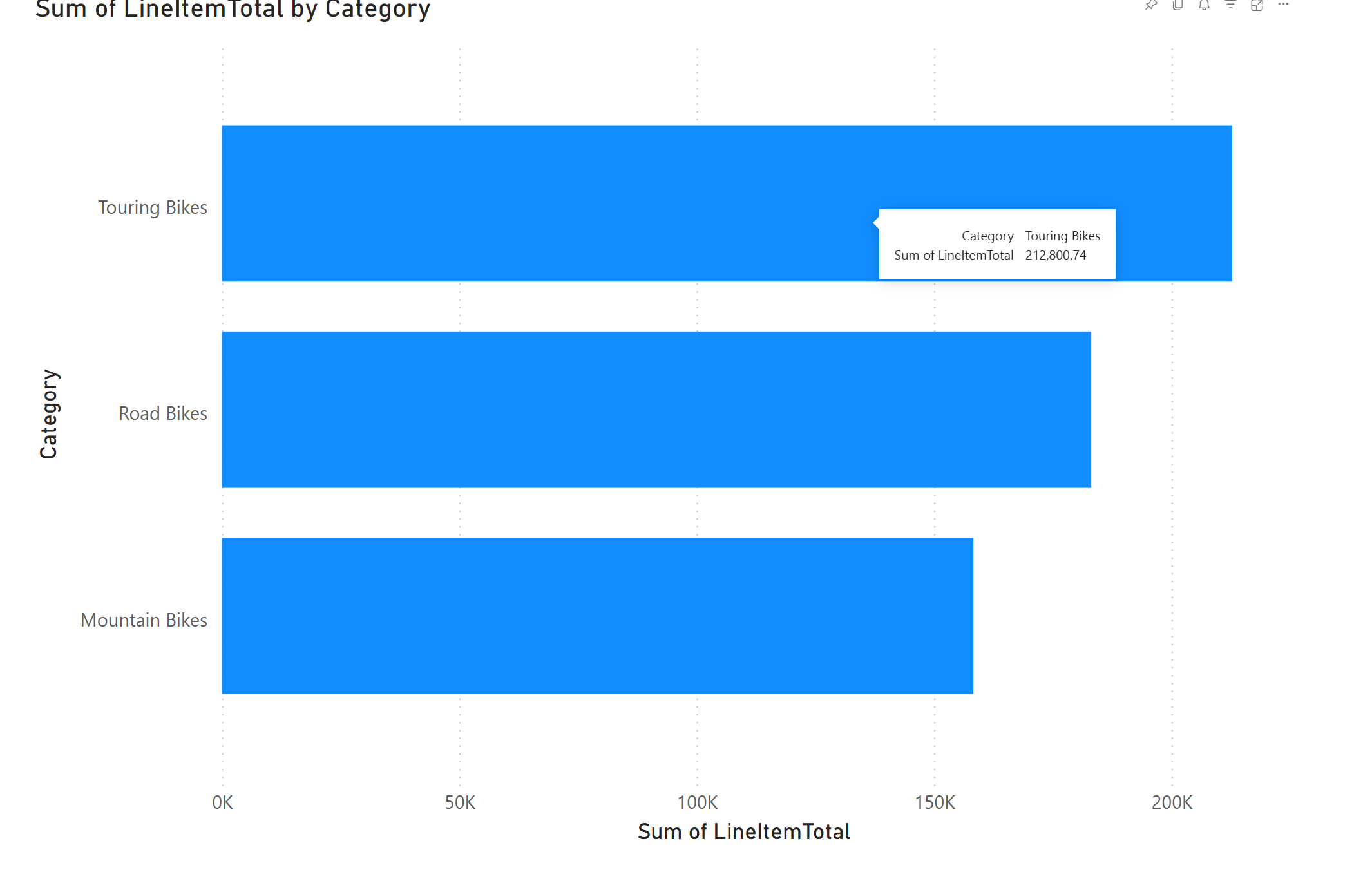
You need to create the following relationships in the default semantic model:

* prod\_customers to prod\_orders
* prod\_products to prod\_orders
* date to prod\_orders

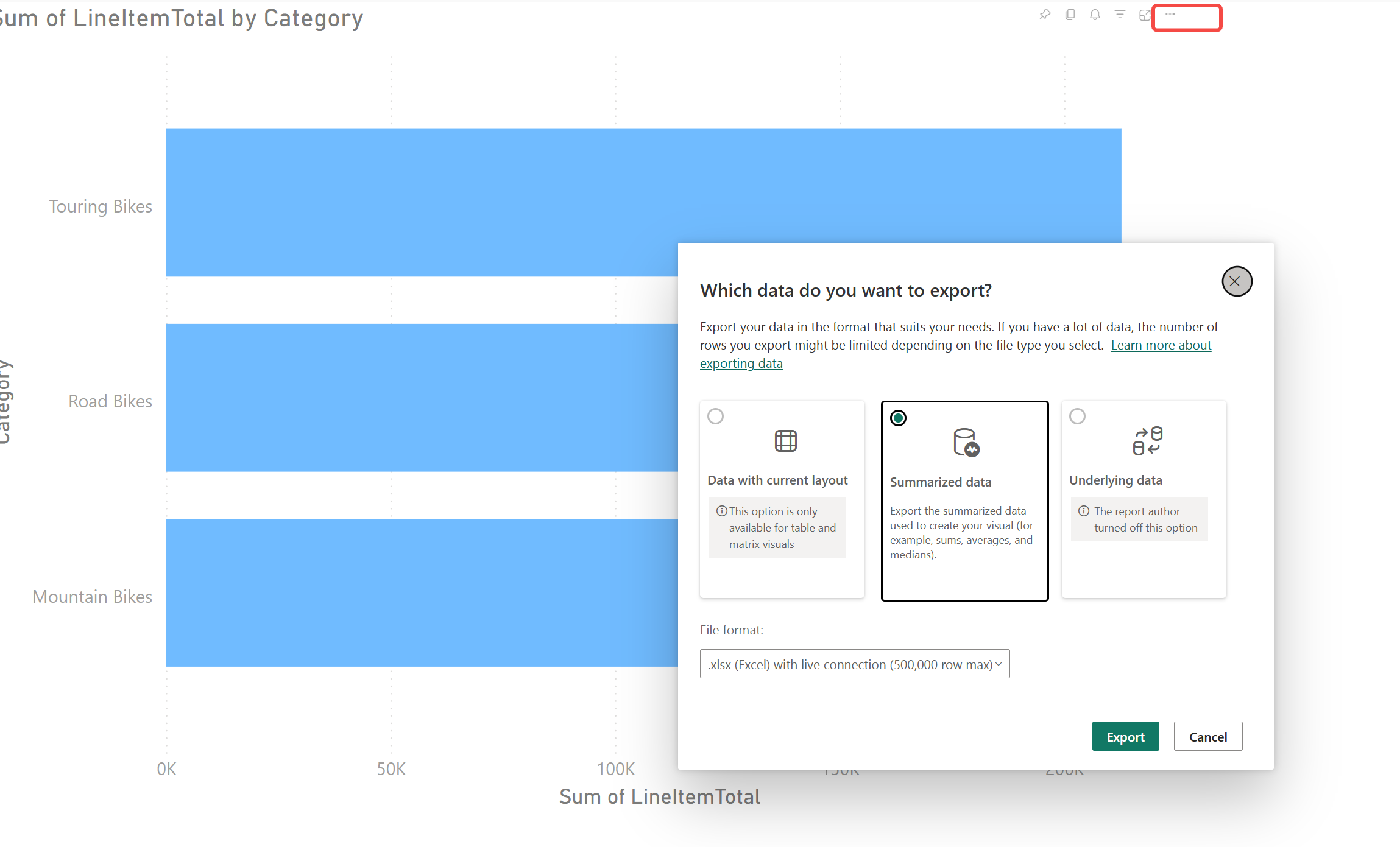


In the bike sales report, create a table visual that meets the following requirements:

* Is filtered to include only the following product categories: **Mountain Bikes**, **Road Bikes**, and **Touring Bikes**
* Shows the sum of the LineItemTotal field by product category



Export the summarized data from the table visual as a CSV file in **C:\files\export\**.



**This is your task:**

First, you need to import the **mynotebook1.ipynb** notebook into the **ws45511000** workspace.

Once the notebook has been uploaded, modify the code to meet the following requirements:

* Create a DataFrame named **df** that lists each CustomerID and the associated average quantity of products ordered and the average cost per product ordered. CustomerID must be the name of the first column. It should have a data type of **varchar(10)**.
* Save the DataFrame to a delta table named **order\_analysis** in the Sales lakehouse.

\*\***Only complete code snippets indicated by # TODO comments.**

Once you have made all of the modifications, run the notebook.