

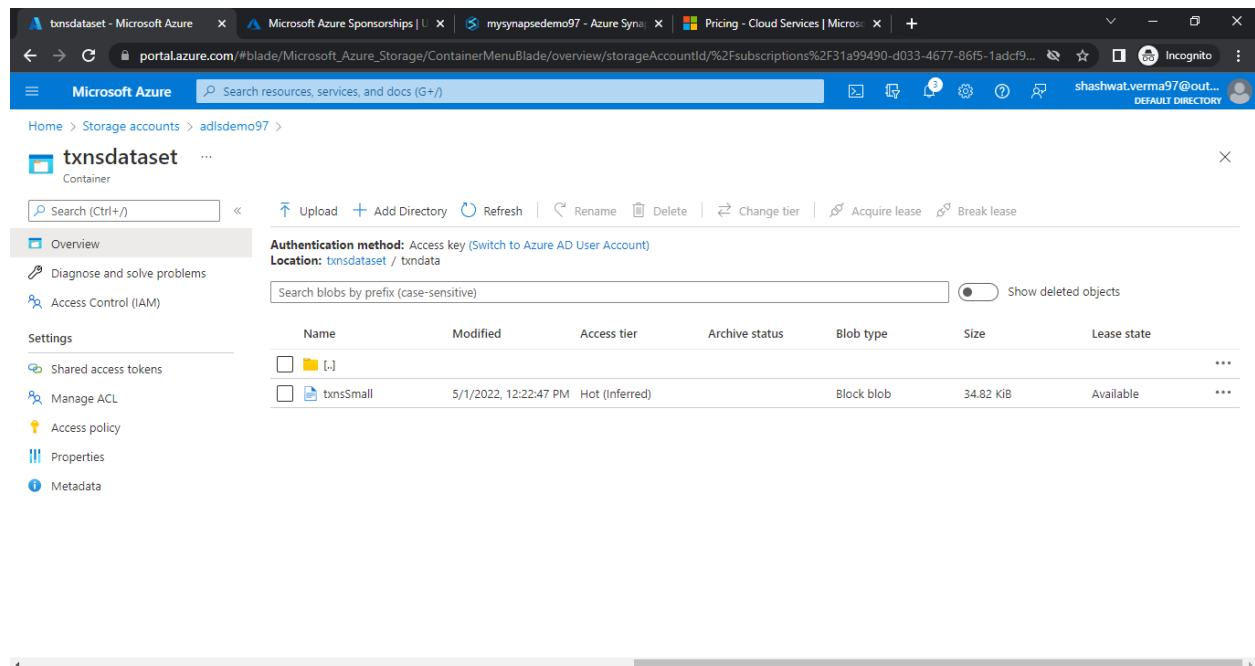
# Concept Assignment 1 : Azure Data Engineering

Perform the following using txns Dataset using

1. SQLPool Internal Table
2. SparkPool
3. PySpark Notebook in Databricks
4. Table Method in Databricks

- a. Find the total revenue generated based on category
- b. Find the total number of transactions done by card
- c. Find the highest selling category
- d. Find the lowest selling category

*In the 'adlsdemo97' storage account we created a container named 'txnsdataset'. Inside txns data set we created a directory 'txndata' where we uploaded 'txnsSmall' file as shown in screenshot below:*



## 1. SQLPool Internal Table

-- Created a separate user for load operations

-- This has to be run in the master database

```

CREATE LOGIN new_user_load WITH PASSWORD = 'Azure@123';

-- SQLPool database
CREATE USER new_user_load FOR LOGIN new_user_load;
GRANT ADMINISTER DATABASE BULK OPERATIONS TO new_user_load;
GRANT CREATE TABLE TO new_user_load;
GRANT ALTER ON SCHEMA::dbo TO new_user_load;

CREATE WORKLOAD GROUP DataLoads
WITH (
    MIN_PERCENTAGE_RESOURCE = 100
    ,CAP_PERCENTAGE_RESOURCE = 100
    ,REQUEST_MIN_RESOURCE_GRANT_PERCENT = 100
);

CREATE WORKLOAD CLASSIFIER [ELTLogin]
WITH (
    WORKLOAD_GROUP = 'DataLoads'
    ,MEMBERNAME = 'new_user_load'
);
-- Create a normal table
-- Login as the new user and create the table
-- Here I have added more constraints when it comes to the width of the data type

CREATE TABLE [txnstable]
(
    [txnId] [bigint] NULL,
    [txnDate] [date] NULL,
    [custid] [bigint] NULL,
    [amount] [float] NULL,
    [category] [varchar](100) NULL,
    [subcategory] [varchar](200) NULL,
    [city] [varchar](100) NULL,
    [state] [varchar](100) NULL,
    [txntype] [varchar](20) NULL)

-- Grant the required privileges to the new user

GRANT INSERT ON txnstable TO new_user_load;
GRANT SELECT ON txnstable TO new_user_load;

```

-- Now log in as the new user

-- The FIRSTROW option helps to ensure the first header row is not part of the COPY implementation

--

<https://docs.microsoft.com/en-us/sql/t-sql/statements/copy-into-transact-sql?view=azure-sql-dw-latest&preserve-view=true>

--<https://adlsdemo97.blob.core.windows.net/txnsdataset/txndata/txnsSmall>

-- Here there is no authentication/authorization, so we need to allow public access for the container

COPY INTO txnstable FROM

'<https://adlsdemo97.blob.core.windows.net/txnsdataset/txndata/txnsSmall>'

WITH

(

FIRSTROW=2)

SELECT \* FROM [txnstable]

SQLQuery1.sql - mysynapsedemo97.sql.azuresynapse.net.mysqlpool97 (sqladminuser (57)) - Microsoft SQL Server Management Studio

Object Explorer

- mysynapsedemo97.sql.azuresynapse.net
  - Databases
    - master
      - Tables
      - Views
      - Synonyms
      - Programmability
      - Security
    - mysqlpool97
      - Tables
      - External Tables
      - Views
      - External Resources
      - Programmability
      - Storage
      - Security
    - Integration Services Catalogs

SQLQuery1.sql - my...sqladminuser (57)\*

```
GRANT INSERT ON txnstable TO new_user_load;
GRANT SELECT ON txnstable TO new_user_load;

-- Now log in as the new user
-- The FIRSTROW option helps to ensure the first header row is not part of the COPY implementation
-- https://docs.microsoft.com/en-us/sql/t-sql/statements/copy-into-transact-sql?view=azure-sql-dw-latest&preserve-view=true

--https://adlsdemo97.blob.core.windows.net/txnsdataset/txndata/txnsSmall
-- Here there is no authentication/authorization, so you need to allow public access for the container
COPY INTO txnstable FROM 'https://adlsdemo97.blob.core.windows.net/txnsdataset/txndata/txnsSmall'
WITH
(
    FIRSTROW=2
)

SELECT * FROM [txnstable]
```

Results

bnid	bnDate	custid	amount	category	subcategory	city	state	bntype
1	2011-03-13	4003268	107.8	Team Sports	Field Hockey	Honolulu	Hawaii	credit
2	2011-02-14	4007591	193.63	Outdoor Recreation	Camping & Backpacking & Hiking	Chicago	Illinois	credit
3	2011-05-26	4006742	198.44	Exercise & Fitness	Weightlifting Gloves	Long Beach	California	credit
4	2011-10-28	4002190	27.89	Puzzles	Jigsaw Puzzles	Charleston	South Carolina	credit
5	2011-06-01	4009775	5.58	Exercise & Fitness	Weightlifting Machine Accessories	Anaheim	California	credit
6	2011-06-05	4002199	198.19	Gymnastics	Gymnastics Rings	Milwaukee	Wisconsin	credit
7	2011-02-25	4004613	36.81	Gymnastics	Vaulting Horses	Los Angeles	California	credit
8	2011-02-08	4002473	41.52	Indoor Games	Bowling	San Francisco	California	credit
9	2011-05-17	4004798	152.46	Jumping	Bungee Jumping	St. Petersburg	Florida	credit
10	2011-01-17	4007361	10.44	Winter Sports	Snowmobiling	Des Moines	Iowa	credit
11	2011-06-18	4008071	121.39	Outdoor Play Equipment	Swing Sets	Columbus	Ohio	credit

Query executed successfully.

mysynapsedemo97.sql.azuresynapse.net sqladminuser (57) mysqlpool97 00:00:00 399 rows

## Problems:

--a. Find the total revenue generated based on category

```
SELECT [category], SUM([amount]) AS revenue FROM [txnstable]
GROUP BY [category]
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
--a. Find the total revenue generated based on category
SELECT [category], SUM([amount]) AS revenue FROM [txnstable]
GROUP BY [category]
```

The Results pane displays the following data:

category	revenue
Team Sports	5178.71
Combat Sports	1655.38
Outdoor Play Equipment	2419.12
Puzzles	508.86
Racquet Sports	1221.73
Water Sports	4379.96
Indoor Games	2498.89
Winter Sports	1314.25
Jumping	2078.58
Gymnastics	3373.12
Air Sports	1086.23

--b. Find the total number of transactions done my card

```
SELECT COUNT(*) as numberofcardtxns FROM [txnstable] WHERE [txntype] = 'credit'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

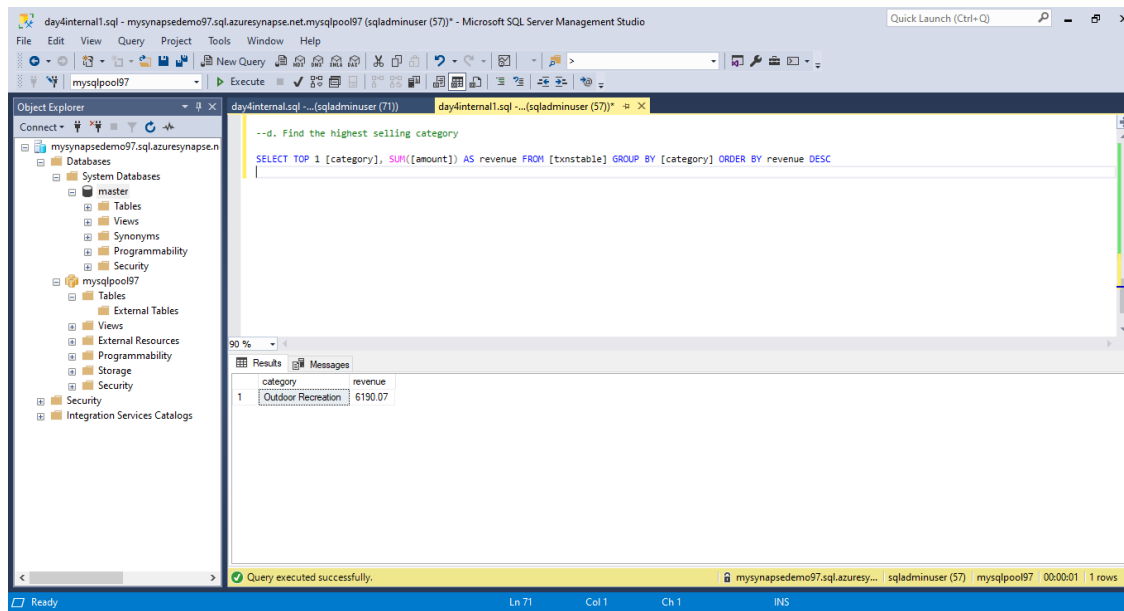
```
--b. Find the total number of transactions done my card
SELECT COUNT(*) as numberofcardtxns FROM [txnstable] WHERE [txntype] = 'credit'
```

The Results pane displays the following data:

numberofcardtxns
335

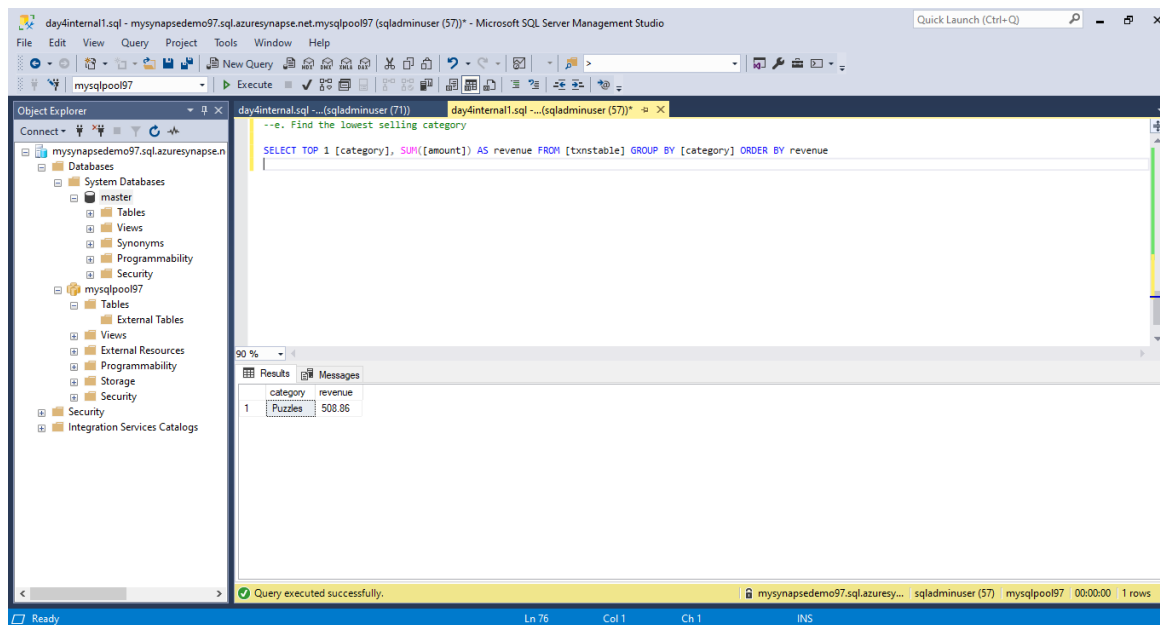
--c. Find the highest selling category

```
SELECT TOP 1 [category], SUM([amount]) AS revenue FROM [txnstable] GROUP BY [category] ORDER BY revenue DESC
```



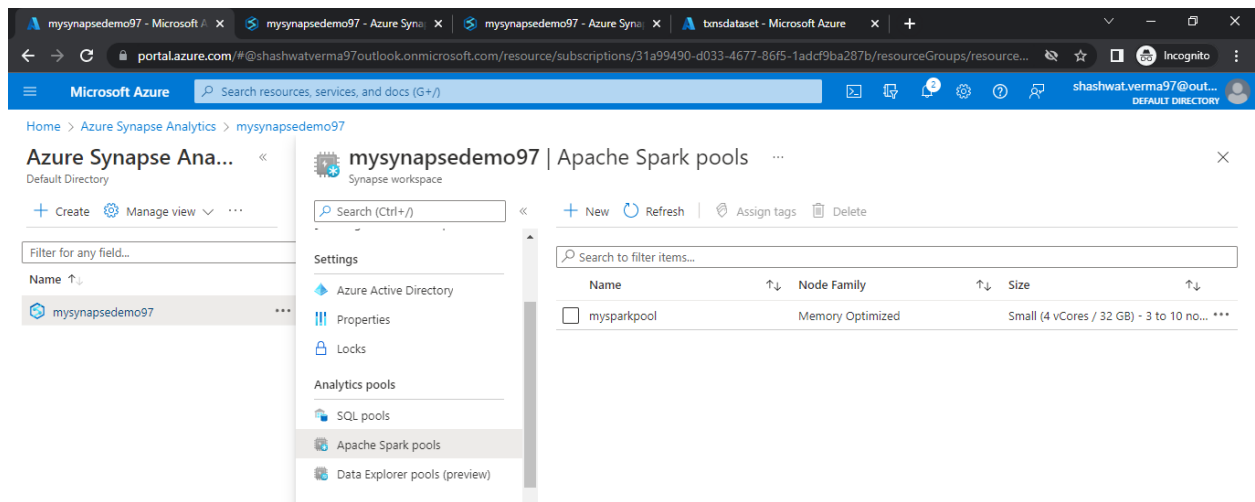
--d. Find the lowest selling category

```
SELECT TOP 1 [category], SUM([amount]) AS revenue FROM [txnstable] GROUP BY [category] ORDER BY revenue
```



## 2. SparkPool

Apache spark pool named 'mysparkpool' was created 'mysynapsedemo97' workspace



### Source Code:

```
#Using Access Key for adls storage
spark.conf.set("fs.azure.account.auth.type.adlsdemo97.dfs.core.window
s.net","SharedKey")
spark.conf.set("fs.azure.account.key.adlsdemo97.dfs.core.windows.net"
,"IgWwisxE9bIfEgv5v52lp5nE9gb2dX8rl9ZsFDiJAXRv4Y2n+IXderAkHeMJHYLbc1h
OFiRPcwZT+AstzQuYeA==")

# Create a Schema Programatically and Assign the same during
DataFrame Creation
from pyspark.sql.types import StructType, StructField
from pyspark.sql.types import
IntegerType, StringType, DoubleType, LongType, DateType

schemaFortxnsData = StructType([
    StructField("txnid", IntegerType(), True),
    StructField("txndate", StringType(), True),
    StructField("custid", LongType(), True),
    StructField("amount", DoubleType(), True),
    StructField("category", StringType(), True),
    StructField("subcategory", StringType(), True),
    StructField("city", StringType(), True),
    StructField("state", StringType(), True),
    StructField("txntype", StringType(), True),
])
```

```

1 #Using Access Key for adls storage
2 spark.conf.set("fs.azure.account.auth.type:adlsdemo97.dfs.core.windows.",
3 spark.conf.set("fs.azure.account.key:adlsdemo97.dfs.core.windows.net",")

[2] ✓ <1 sec - Command executed in 151 ms by shashwat.verma97 on 8:44:16 PM, 5/05/22

1 from pyspark.sql.types import StructType,StructField
2 from pyspark.sql.types import IntegerType,StringType,DoubleType,LongType,DateType
3
4 schemaFortxnsData = StructType([
5     StructField("txnid", IntegerType(),True),
6     StructField("txndate", StringType(),True),
7     StructField("custid", LongType(),True),
8     StructField("amount", DoubleType(),True),
9     StructField("category", StringType(),True),
10    StructField("subcategory", StringType(),True),
11    StructField("city", StringType(),True),
12    StructField("state", StringType(),True),
13    StructField("txntype", StringType(),True),
14 ])

[4] ✓ <1 sec - Command executed in 163 ms by shashwat.verma97 on 8:47:26 PM, 5/05/22

```

```
# Assigning the Schema
```

```
txnsDataDF =
```

```
spark.read.schema(schemaFortxnsData).option("delimiter",',' ).csv('abfss://txnsdataset@adlsdemo97.dfs.core.windows.net/txndata/txnsSmall')
```

```
txnsDataDF.show(5)
```

```

1 # Assigning the Schema
2 txnsDataDF = spark.read.schema(schemaFortxnsData).option("delimiter",',' ).csv('abfss://txnsdataset@adlsdemo97.dfs.c

[5] ✓ 2 sec - Command executed in 2 sec 100 ms by shashwat.verma97 on 8:50:56 PM, 5/05/22

1 txnsDataDF.show(5)

[6] ✓ 7 sec - Command executed in 6 sec 923 ms by shashwat.verma97 on 8:51:22 PM, 5/05/22

```

Job execution Succeeded Spark 2 executors 8 cores

txnid	txndate	custid	amount	category	subcategory	city	state	txntype
0 06-26-2011 4007024	40.33 Exercise & Fitness Cardio Machine Ac...	Clarksville	Tennessee	credit				
1 05-26-2011 4006742	198.44 Exercise & Fitness Weightlifting Gloves	Long Beach	California	credit				
2 06-01-2011 4009775	5.58 Exercise & Fitness Weightlifting Mac...	Anaheim	California	credit				
3 06-05-2011 4002199	198.19 Gymnastics Gymnastics Rings	Milwaukee	Wisconsin	credit				
4 12-17-2011 4002613	98.81 Team Sports Field Hockey	Nashville	Tennessee	credit				

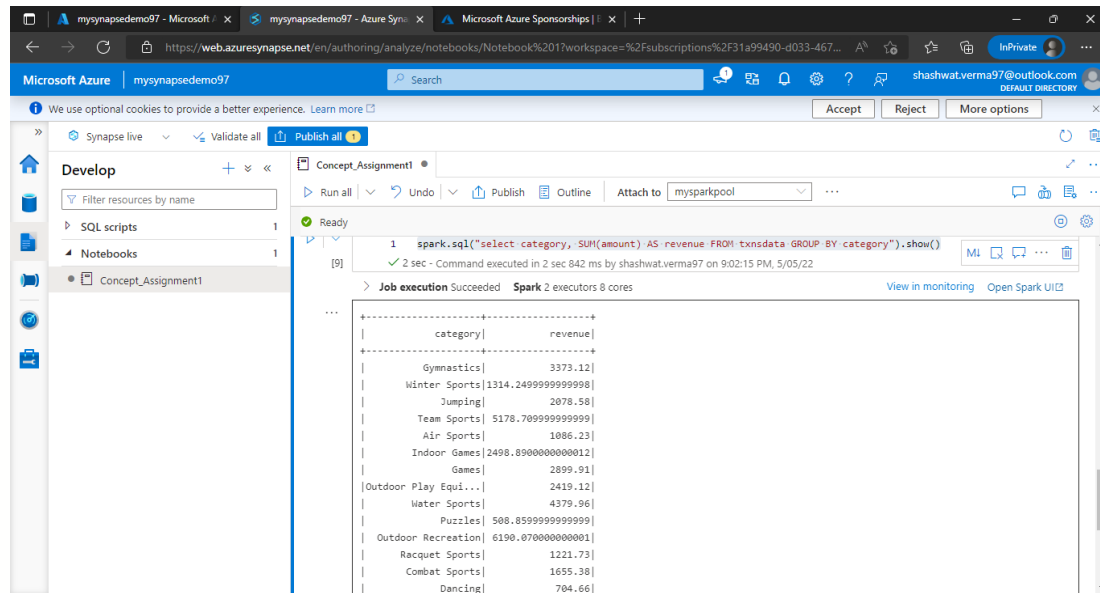
```
# Register this DataFrame as a Temporary Table in Apache Spark
```

```
txnsDataDF.registerTempTable("txnsdata")
```

## Problems:

a. Find the total revenue generated based on category

```
spark.sql("select category, SUM(amount) AS revenue FROM txnsdata  
GROUP BY category").show()
```

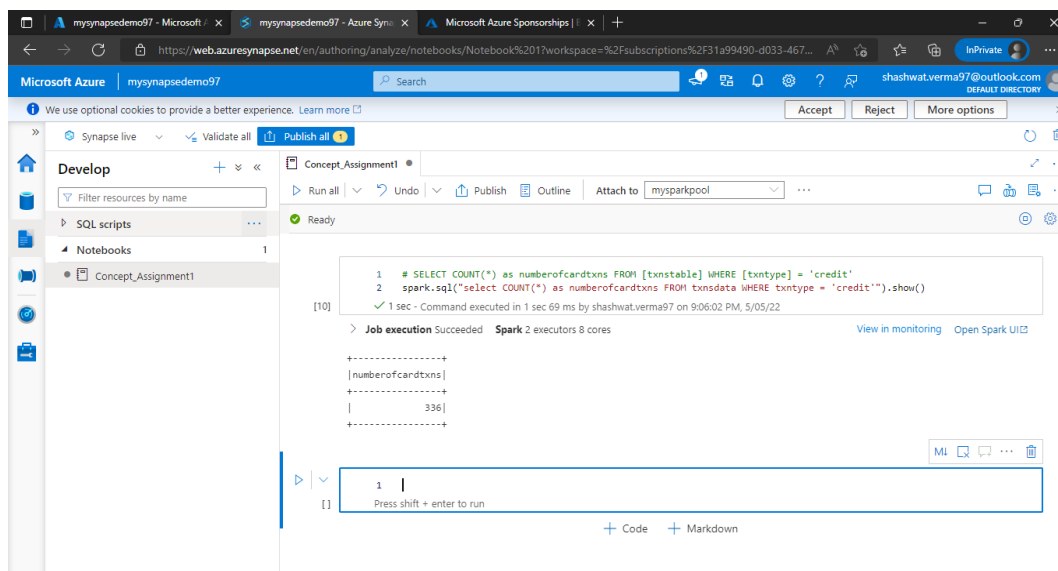


Job execution Succeeded Spark 2 executors 8 cores

category	revenue
Gymnastics	3373.12
Winter Sports	1314.2499999999998
Jumping	2878.58
Team Sports	5178.7099999999999
Air Sports	1086.23
Indoor Games	2498.8900000000001
Games	2899.91
Outdoor Play Equi...	2419.12
Water Sports	4379.96
Puzzles	508.85999999999999
Outdoor Recreation	6190.0700000000001
Racquet Sports	1221.73
Combat Sports	1655.38
Dancing	704.66

b. Find the total number of transactions done my card

```
spark.sql("select COUNT(*) as numberofcardtxns FROM txnsdata WHERE  
txntype = 'credit'").show()
```



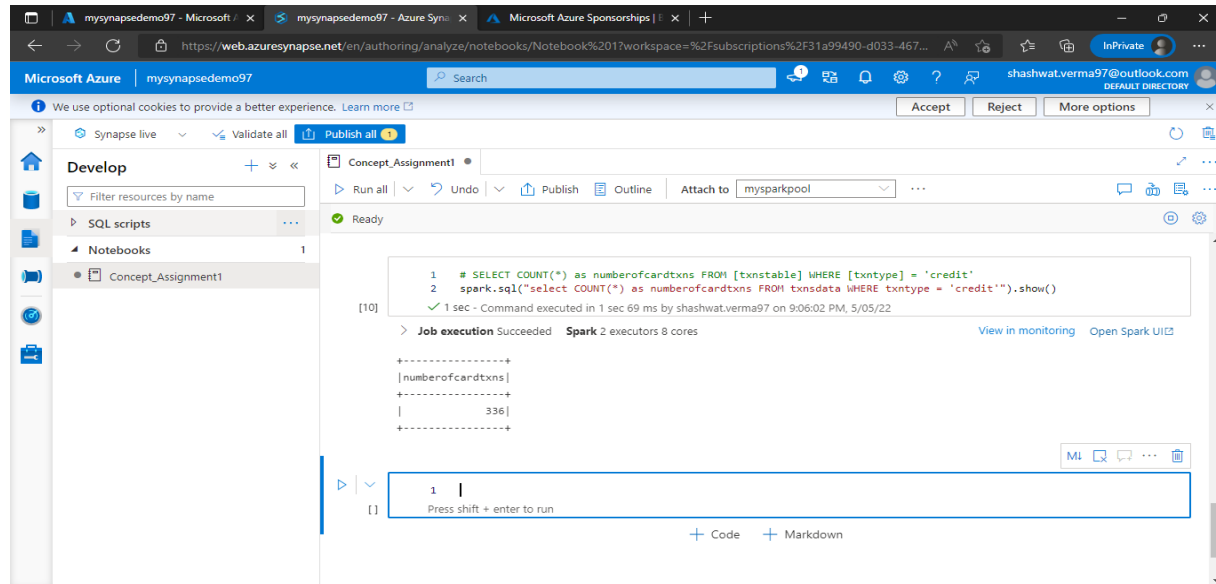
Job execution Succeeded Spark 2 executors 8 cores

numberofcardtxns
336



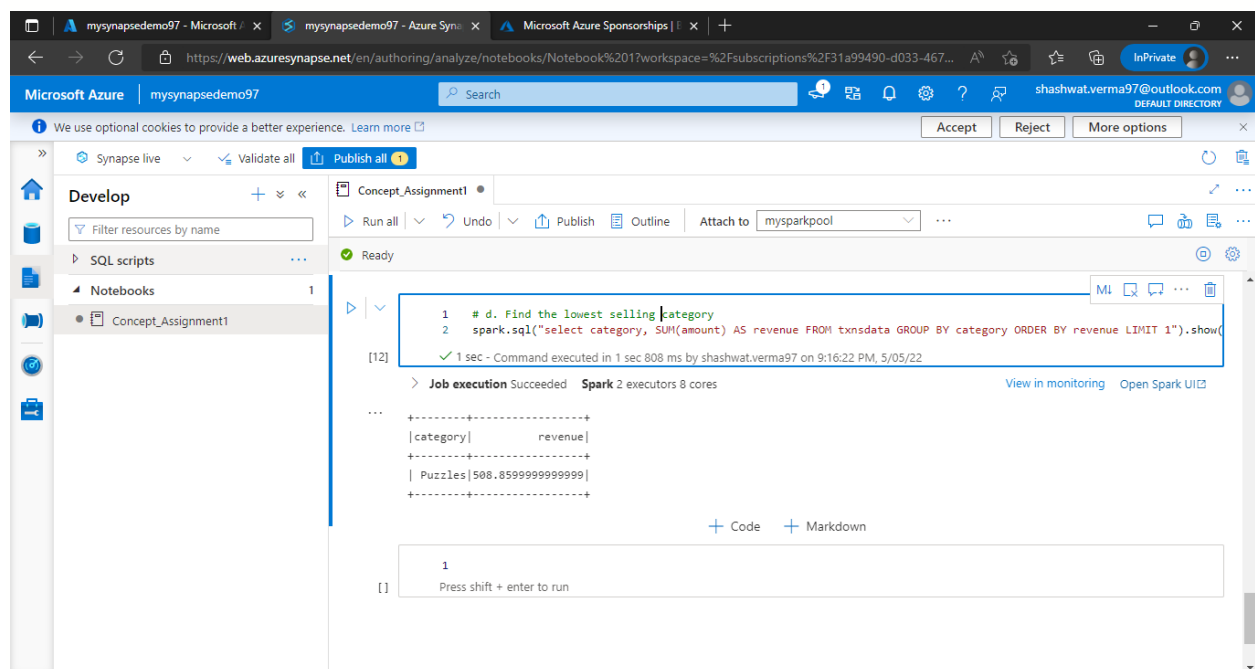
### c. Find the highest selling category

```
spark.sql("select category, SUM(amount) AS revenue FROM txnsdata  
GROUP BY category ORDER BY revenue DESC LIMIT 1").show()
```



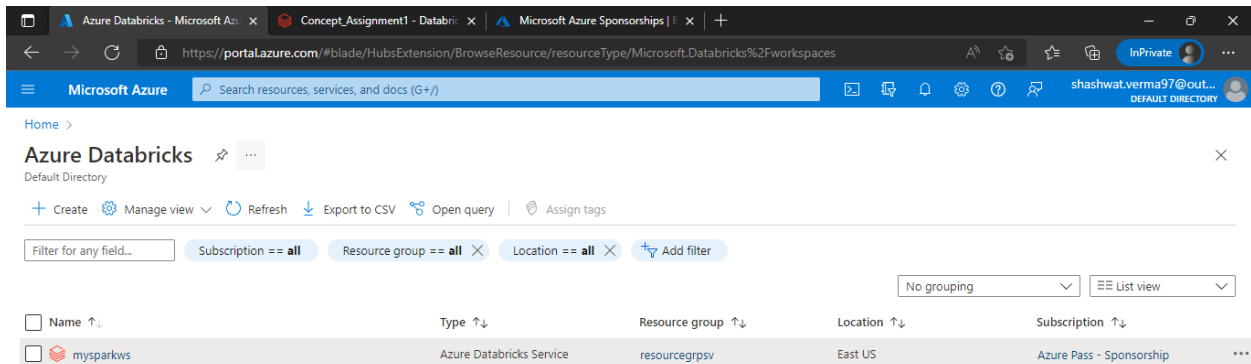
### d. Find the lowest selling category

```
spark.sql("select category, SUM(amount) AS revenue FROM txnsdata  
GROUP BY category ORDER BY revenue LIMIT 1").show()
```



### 3. PySpark Notebook in Databricks

We create a Azure Databricks workspace 'mysparkws' which is launched and created a cluster 'mytestcluster9706' inside which we created a notebook 'Concept\_Assignment1'



### Source Code:

#Using Access Key for adls storage

```
spark.conf.set("fs.azure.account.auth.type.adlsdemo97.dfs.core.windows.net","SharedKey")
spark.conf.set("fs.azure.account.key.adlsdemo97.dfs.core.windows.net","lgWwisxE9bIfEgv5v52
1p5nE9gb2dX8rl9ZsFDiJAXRv4Y2n+IXderAkHeMJHYLbc1hOFiRPcwZT+ASztQuYeA==")
```

# Create a Schema Programatically and Assign the same during DataFrame Creation

```
from pyspark.sql.types import StructType, StructField
```

```
from pyspark.sql.types import IntegerType, StringType, DoubleType, LongType, DateType
```

```
schemaFortxnsData = StructType([
    StructField("txnid", IntegerType(), True),
    StructField("txndate", StringType(), True),
    StructField("custid", LongType(), True),
    StructField("amount", DoubleType(), True),
    StructField("category", StringType(), True),
    StructField("subcategory", StringType(), True),
    StructField("city", StringType(), True),
    StructField("state", StringType(), True),
    StructField("txntype", StringType(), True),
])
```

# Assigning the Schema

```
txnsDataDF =
```

```
spark.read.schema(schemaFortxnsData).option("delimiter",',').csv('abfss://txnsdataset@adlsdemo97.dfs.core.windows.net/txndata/txnsSmall')
```

```

1 #Using Access Key for adls storage
2 spark.conf.set("fs.azure.account.auth.type.adlsdemo97.dfs.core.windows.net", "SharedKey")
3 spark.conf.set("fs.azure.account.key.adlsdemo97.dfs.core.windows.net", "IgWw1sxE9bIFegv5v52Ip5nE9gb2dX8rL9ZsFD1JAXRv4Y2n+IXderAkHeMJHYLbc1h0FiRcPwZT+ASTzQuYeA==")

```

Command took 0.04 seconds -- by shashwat.verma97@outlook.com at 5/5/2022, 10:43:23 PM on mytestcluster9786

```

1 # Create a Schema Programmatically and Assign the same during DataFrame Creation
2 from pyspark.sql.types import StructType, StructField
3 from pyspark.sql.types import IntegerType, StringType, DoubleType, LongType, DateType
4
5 schemaFortxnsData = StructType([
6     StructField("txnid", IntegerType(), True),
7     StructField("txndate", StringType(), True),
8     StructField("custid", LongType(), True),
9     StructField("amount", DoubleType(), True),
10    StructField("category", StringType(), True),
11    StructField("subcategory", StringType(), True),
12    StructField("city", StringType(), True),
13    StructField("state", StringType(), True),
14    StructField("txntype", StringType(), True),
15 ])

```

Command took 0.07 seconds -- by shashwat.verma97@outlook.com at 5/5/2022, 10:43:37 PM on mytestcluster9786

# Assigning the Schema

txnsDataDF =

```
spark.read.schema(schemaFortxnsData).option("delimiter",',' ).csv('abfss://txnsdataset@adlsdemo97.dfs.core.windows.net/txndata/txnsSmall')
```

txnsDataDF.show(5)

```

1 # Assigning the Schema
2 txnsDataDF = spark.read.schema(schemaFortxnsData).option("delimiter",',' ).csv('abfss://txnsdataset@adlsdemo97.dfs.core.windows.net/txndata/txnsSmall')

```

txnsDataDF: pyspark.sql.dataframe.DataFrame = [txnid: integer, txndate: string ... 7 more fields]

Command took 1.58 seconds -- by shashwat.verma97@outlook.com at 5/5/2022, 10:43:56 PM on mytestcluster9786

```

1 txnsDataDF.show(5)

```

(1) Spark Jobs

txnid	txndate	custid	amount	category	subcategory	city	state	txntype
0 06-26-2011 4007024	40.33	Exercise & Fitness	Cardio Machine Ac...	Clarksville	Tennessee	credit		
1 05-26-2011 4006742	198.44	Exercise & Fitness	Weightlifting Gloves	Long Beach	California	credit		
2 06-01-2011 4009775	5.58	Exercise & Fitness	Weightlifting Mac...	Anaheim	California	credit		
3 06-05-2011 4002199	198.19	Gymnastics	Gymnastics Rings	Milwaukee	Wisconsin	credit		
4 12-17-2011 4002613	98.81	Team Sports	Field Hockey	Nashville	Tennessee	credit		

only showing top 5 rows

Command took 2.24 seconds -- by shashwat.verma97@outlook.com at 5/5/2022, 10:51:05 PM on mytestcluster9786

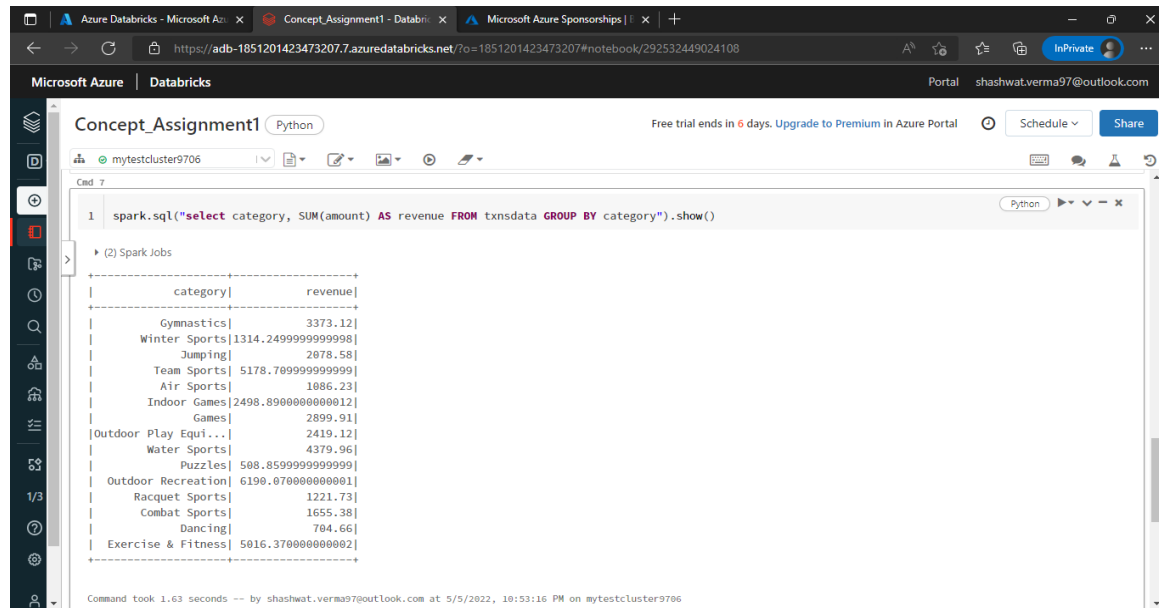
# Register this DataFrame as a Temporary Table in Apache Spark

```
txnsDataDF.registerTempTable("txnsdata")
```

## Problems:

a. Find the total revenue generated based on category

```
spark.sql("select category, SUM(amount) AS revenue FROM txnsdata GROUP BY category").show()
```



The screenshot shows a Databricks notebook with the following content:

```
1 spark.sql("select category, SUM(amount) AS revenue FROM txnsdata GROUP BY category").show()
```

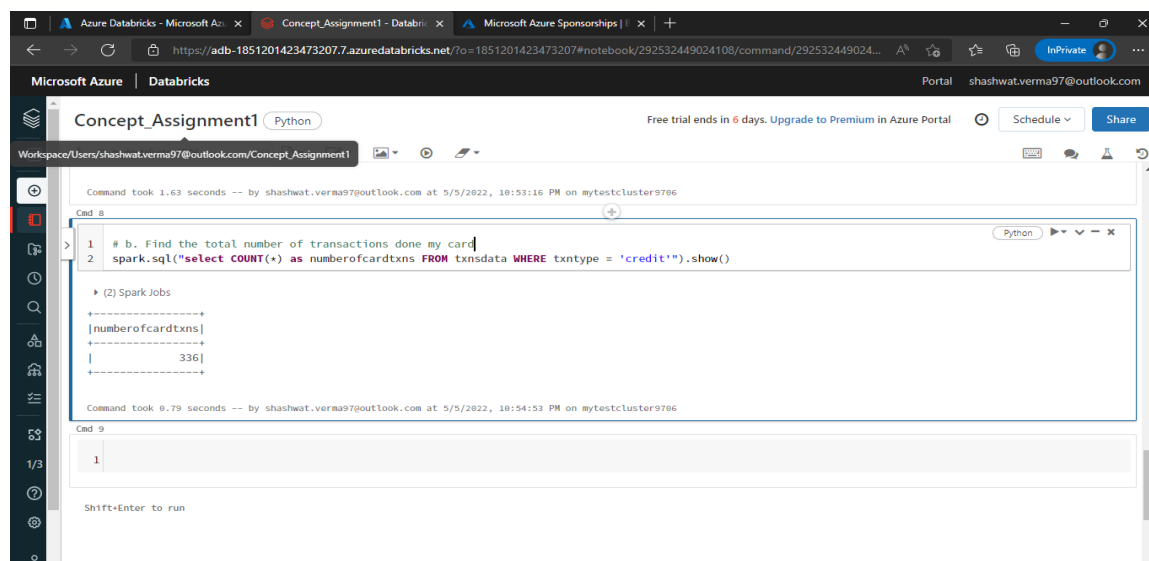
The output is a table with two columns: category and revenue.

category	revenue
Gymnastics	3373.12
Winter Sports	1314.2499999999998
Jumping	2878.58
Team Sports	5178.7899999999999
Air Sports	1886.23
Indoor Games	2498.8900000000001
Games	2899.91
Outdoor Play Equi...	2419.12
Water Sports	4379.96
Puzzles	508.85999999999999
Outdoor Recreation	6190.8700000000001
Racquet Sports	1221.73
Combat Sports	1655.38
Dancing	784.66
Exercise & Fitness	5816.3700000000002

Command took 1.63 seconds -- by shashwat.verma97@outlook.com at 5/5/2022, 10:53:16 PM on mytestcluster9706

b. Find the total number of transactions done my card

```
spark.sql("select COUNT(*) as numberofcardtxns FROM txnsdata WHERE txntype = 'credit'").show()
```



The screenshot shows a Databricks notebook with the following content:

```
1 # b. Find the total number of transactions done my card
2 spark.sql("select COUNT(*) as numberofcardtxns FROM txnsdata WHERE txntype = 'credit'").show()
```

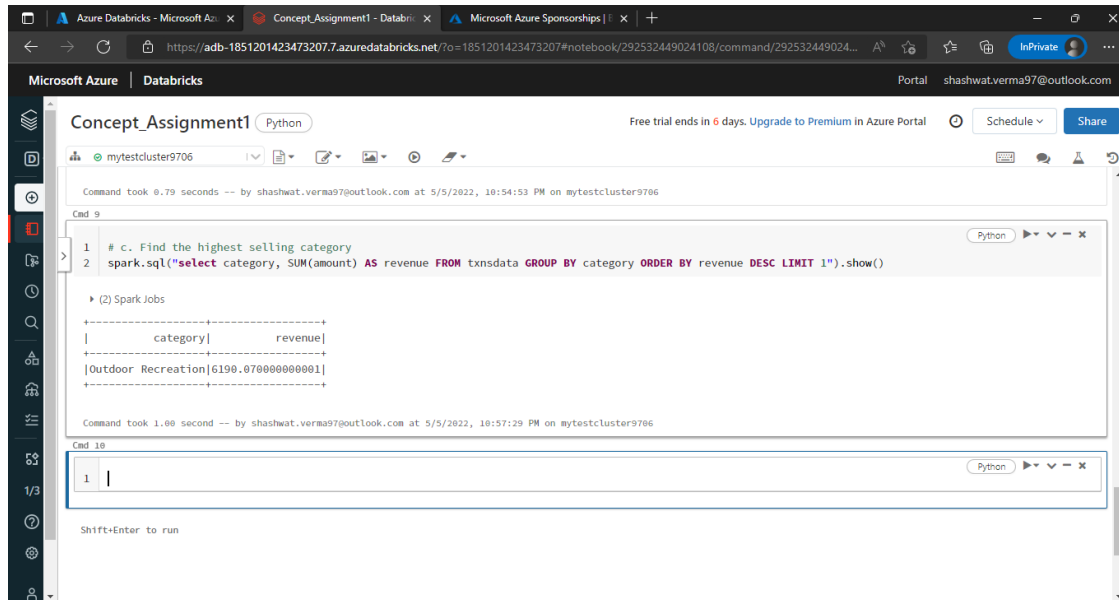
The output is a table with one column: numberofcardtxns.

numberofcardtxns
336

Command took 0.79 seconds -- by shashwat.verma97@outlook.com at 5/5/2022, 10:54:53 PM on mytestcluster9706

c. Find the highest selling category

```
spark.sql("select category, SUM(amount) AS revenue FROM txnsdata GROUP BY category ORDER BY revenue DESC LIMIT 1").show()
```



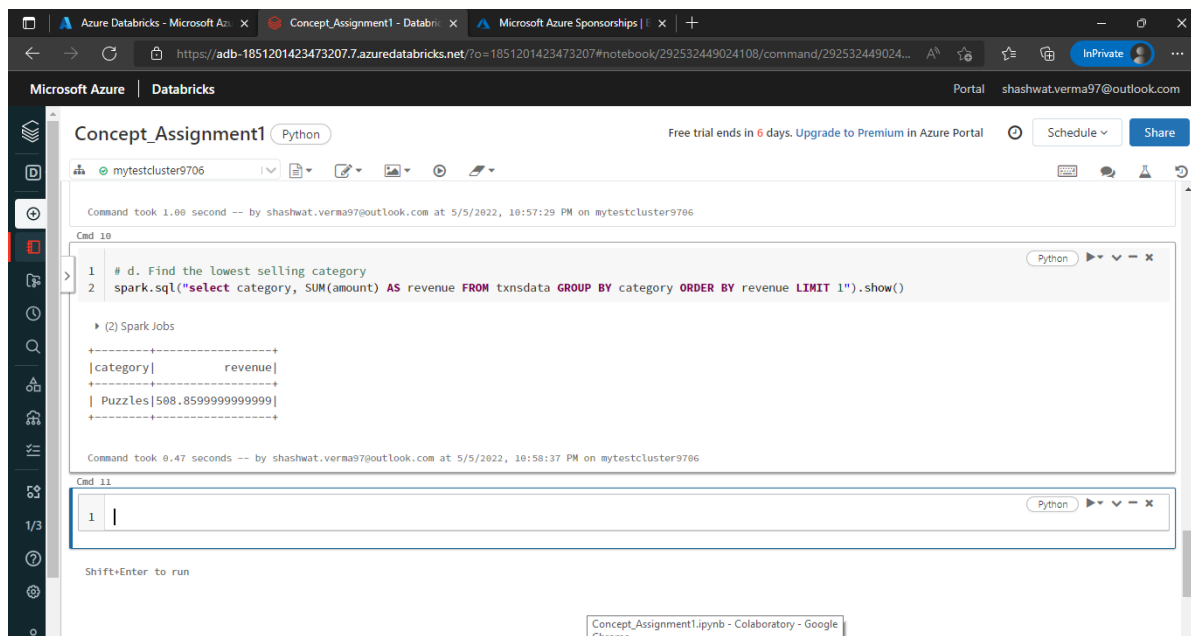
The screenshot shows a Databricks notebook interface. The top bar indicates the user is logged in as 'shashwat.verma97@outlook.com'. The notebook is titled 'Concept\_Assignment1' and is in 'Python' mode. The command history shows a previous command that took 0.79 seconds. The current command (Cmd 9) is a Spark SQL query to find the highest selling category. The output shows a single row for 'Outdoor Recreation' with a revenue of 6190.070000000001.

```
1 # c. Find the highest selling category
2 spark.sql("select category, SUM(amount) AS revenue FROM txnsdata GROUP BY category ORDER BY revenue DESC LIMIT 1").show()
```

category	revenue
Outdoor Recreation	6190.070000000001

d. Find the lowest selling category

```
spark.sql("select category, SUM(amount) AS revenue FROM txnsdata GROUP BY category ORDER BY revenue LIMIT 1").show()
```



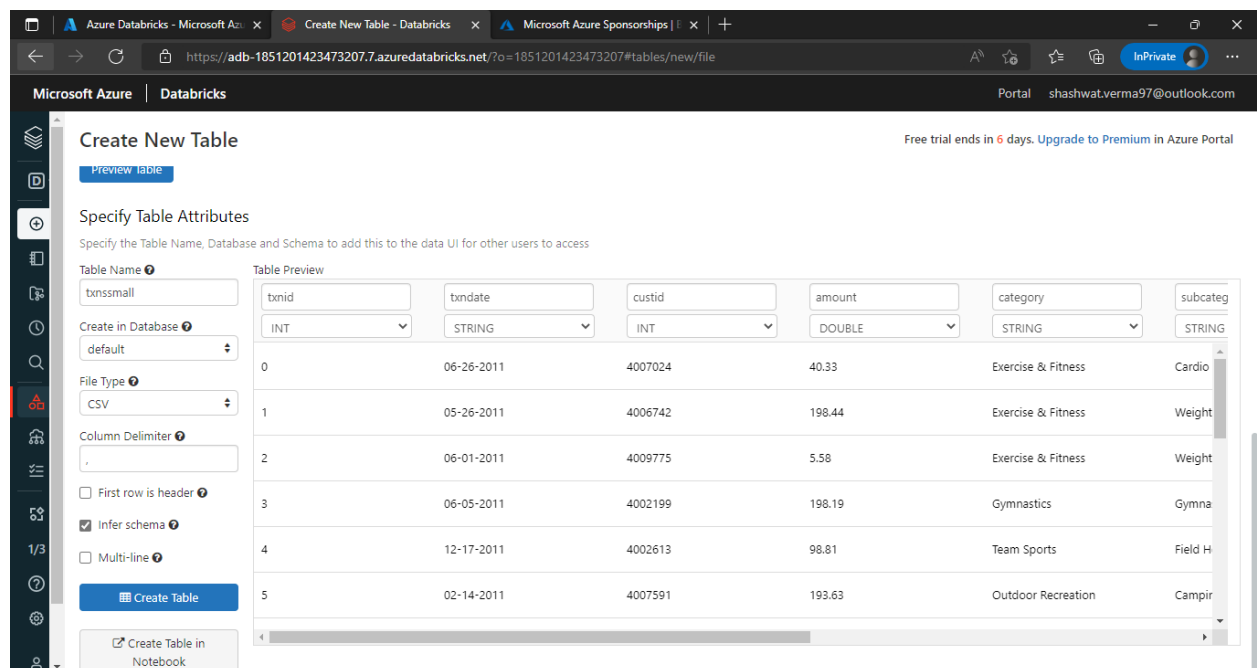
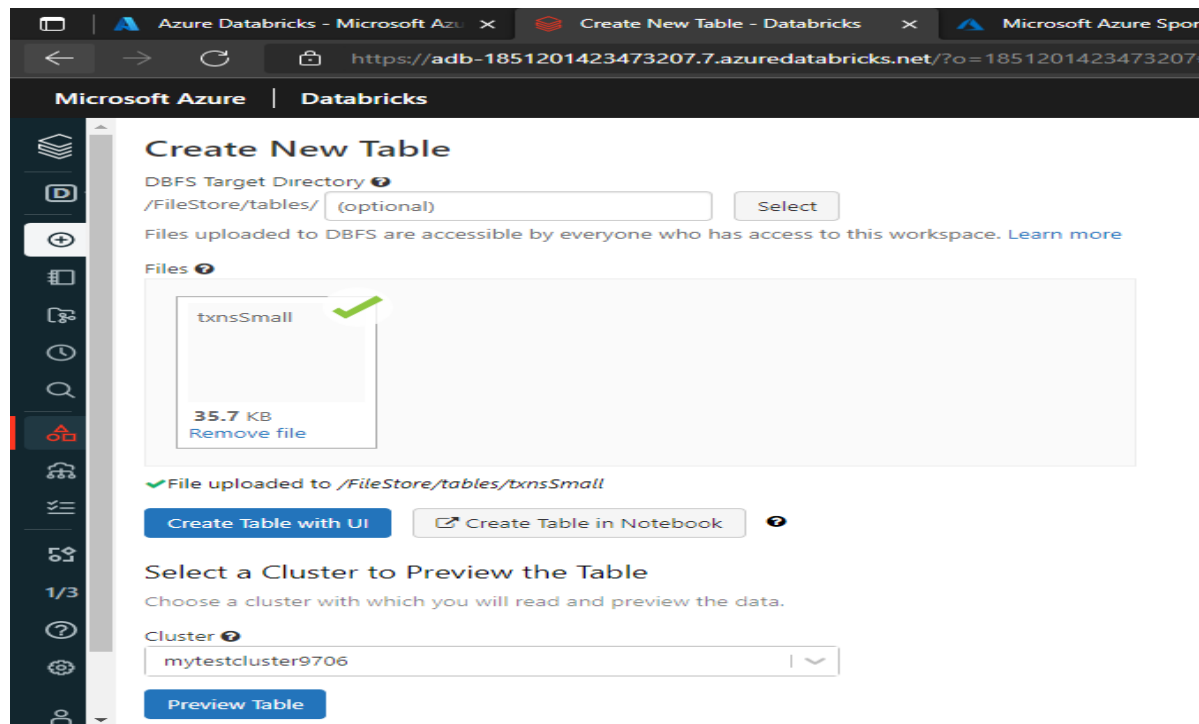
The screenshot shows the same Databricks notebook interface. The current command (Cmd 10) is a Spark SQL query to find the lowest selling category. The output shows a single row for 'Puzzles' with a revenue of 508.8599999999999.

```
1 # d. Find the lowest selling category
2 spark.sql("select category, SUM(amount) AS revenue FROM txnsdata GROUP BY category ORDER BY revenue LIMIT 1").show()
```

category	revenue
Puzzles	508.8599999999999

## 4. Table Method in Databricks

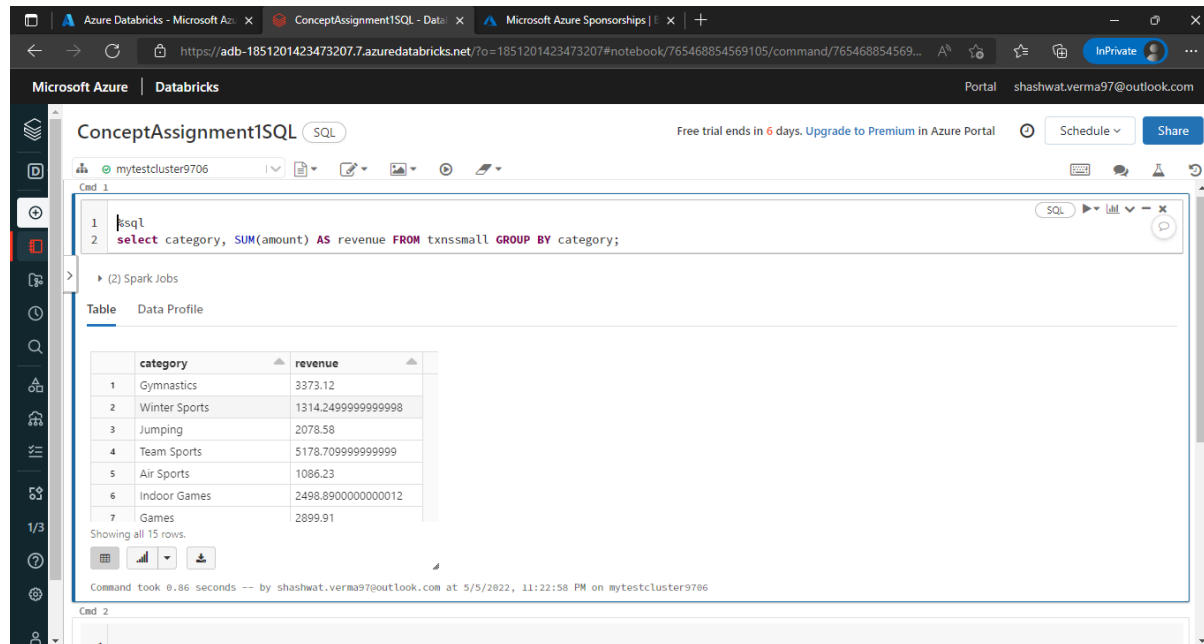
We created a Table using UI inside 'mytestcluster9706' and directly uploaded the file 'txnsSmall'



After hitting create table button for 'txnsSmall' we created a new notebook 'ConceptAssignment1SQL' to perform the tasks

a. Find the total revenue generated based on category

select category, SUM(amount) AS revenue FROM txnssmall GROUP BY category;

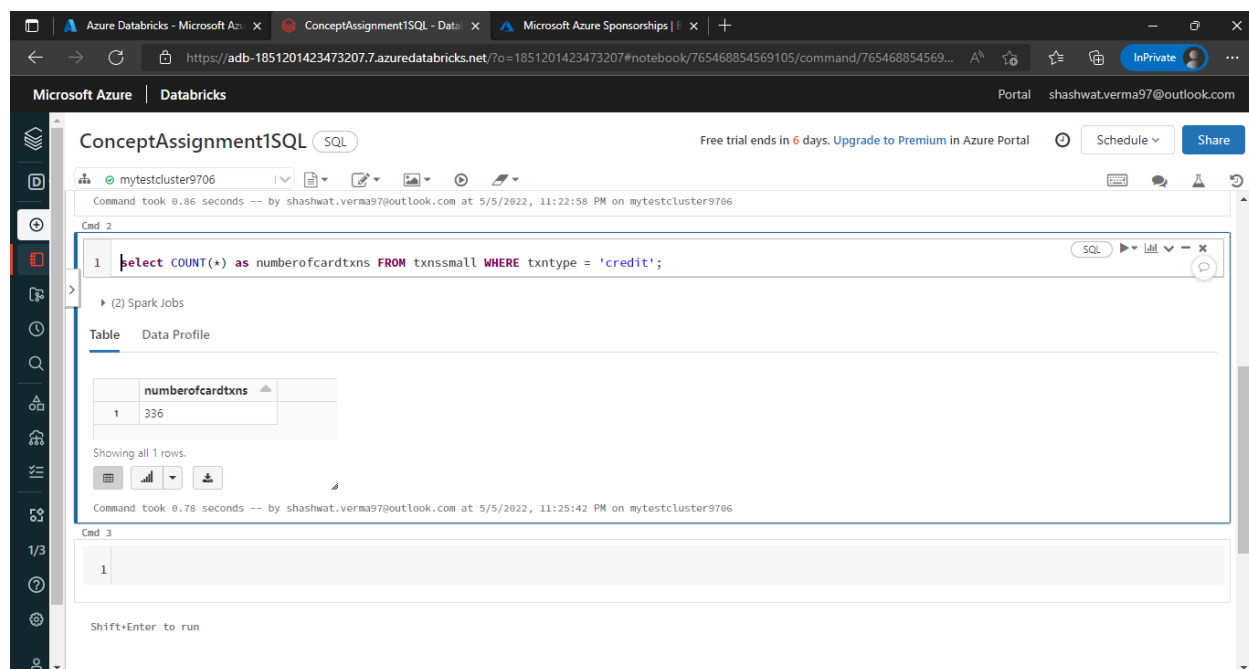


The screenshot shows the Azure Databricks SQL interface. The query executed is: `select category, SUM(amount) AS revenue FROM txnssmall GROUP BY category;`. The result is displayed in a table with 7 rows and 2 columns: category and revenue.

	category	revenue
1	Gymnastics	3373.12
2	Winter Sports	1314.2499999999998
3	Jumping	2078.58
4	Team Sports	5178.7099999999999
5	Air Sports	1086.23
6	Indoor Games	2498.89000000000012
7	Games	2899.91

b. Find the total number of transactions done my card

select COUNT(\*) as numberofcardtxns FROM txnssmall WHERE txntype = 'credit';



The screenshot shows the Azure Databricks SQL interface. The query executed is: `select COUNT(*) as numberofcardtxns FROM txnssmall WHERE txntype = 'credit';`. The result is displayed in a table with 1 row and 2 columns: numberofcardtxns.

	numberofcardtxns
1	336

c. Find the highest selling category

select category, SUM(amount) AS revenue FROM txnssmall GROUP BY category  
ORDER BY revenue DESC LIMIT 1;

The screenshot shows the Azure Databricks SQL interface. The command bar contains the SQL query: `select category, SUM(amount) AS revenue FROM txnssmall GROUP BY category ORDER BY revenue DESC LIMIT 1;`. Below the command bar, the results are displayed in a table with two columns: **category** and **revenue**. The table shows one row: **Outdoor Recreation** with a revenue of **6190.070000000001**. The interface also shows the command execution time and the user's profile.

category	revenue
Outdoor Recreation	6190.070000000001

d. Find the lowest selling category

select category, SUM(amount) AS revenue FROM txnssmall GROUP BY category  
ORDER BY revenue LIMIT 1;

The screenshot shows the Azure Databricks SQL interface. The command bar contains the SQL query: `select category, SUM(amount) AS revenue FROM txnssmall GROUP BY category ORDER BY revenue LIMIT 1;`. Below the command bar, the results are displayed in a table with two columns: **category** and **revenue**. The table shows one row: **Puzzles** with a revenue of **508.8599999999999**. The interface also shows the command execution time and the user's profile.

category	revenue
Puzzles	508.8599999999999