databricks Customer Act Analysis 2024-11-10 18:36:24

(https://databricks.com)

1

```
storage_account_name = "ncplstorageact"
container_name = "raw"
```

2

```
# Define your storage account and container details
storage_account_name = "ncplstorageact"
container_name = "raw"
mount_point = f"/mnt/{container_name}"

# Check if the container is already mounted
if not any(mount.mountPoint == mount_point for mount in dbutils.fs.mounts()):
    dbutils.fs.mount(
        source=f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net",
        mount_point=mount_point,
        extra_configs={
            f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net": dbutils.secrets.get
            ('testScope', 'secret-ncplstorageact')
        }
    )
```

3

```
# Verify the mount
display(dbutils.fs.ls(mount_point))
```

Table

	path	name	size	modificationTime
1	dbfs:/mnt/raw/accounts.csv	accounts.csv	2331	1731284354000
2	dbfs:/mnt/raw/customers.csv	customers.csv	4603	1731284354000
3	dbfs:/mnt/raw/loan_payments.c...	loan_payments.c...	2613	1731284356000
4	dbfs:/mnt/raw/loans.csv	loans.csv	2340	1731284354000
5	dbfs:/mnt/raw/transactions.csv	transactions.csv	3513	1731284354000

5 rows

4

```
# Define the file paths
accounts_path = "/mnt/raw/accounts.csv"
customers_path = "/mnt/raw/customers.csv"
loan_payments_path = "/mnt/raw/loan_payments.csv"
loans_path = "/mnt/raw/loans.csv"
transactions_path = "/mnt/raw/transactions.csv"

# Read each CSV file into a DataFrame
accounts_df = spark.read.csv(accounts_path, header=True, inferSchema=True)
customers_df = spark.read.csv(customers_path, header=True, inferSchema=True)
loan_payments_df = spark.read.csv(loan_payments_path, header=True, inferSchema=True)
loans_df = spark.read.csv(loans_path, header=True, inferSchema=True)
transactions_df = spark.read.csv(transactions_path, header=True, inferSchema=True)
```

5

```
# Get the number of rows in the DataFrame
row_count = accounts_df.count()
row_count = customers_df.count()
row_count = loan_payments_df.count()
row_count = loans_df.count()
row_count = transactions_df.count()

# Display the number of rows
print(f"The number of rows in customers.csv is: {row_count}")

print(f"The number of rows in accounts.csv is: {row_count}")

print(f"The number of rows in loan_payments.csv is: {row_count}")

print(f"The number of rows in loans.csv is: {row_count}")

print(f"The number of rows in transactions.csv is: {row_count}")
```

```
The number of rows in customers.csv is: 100
The number of rows in accounts.csv is: 100
The number of rows in loan_payments.csv is: 100
The number of rows in loans.csv is: 100
The number of rows in transactions.csv is: 100
```

6

```
# Remove rows with null values
accounts_df = accounts_df.dropna()
customers_df = customers_df.dropna()
loan_payments_df = loan_payments_df.dropna()
loans_df = loans_df.dropna()
transactions_df = transactions_df.dropna()
```

7

```
# Get the number of rows in the DataFrame
row_count = accounts_df.count()
row_count = customers_df.count()
row_count = loan_payments_df.count()
row_count = loans_df.count()
row_count = transactions_df.count()

# Display the number of rows
print(f"The number of rows in customers.csv is: {row_count}")

print(f"The number of rows in accounts.csv is: {row_count}")

print(f"The number of rows in loan_payments.csv is: {row_count}")

print(f"The number of rows in loans.csv is: {row_count}")

print(f"The number of rows in transactions.csv is: {row_count}")
```

```
The number of rows in customers.csv is: 100
The number of rows in accounts.csv is: 100
The number of rows in loan_payments.csv is: 100
The number of rows in loans.csv is: 100
The number of rows in transactions.csv is: 100
```

Data Cleaning

""" Check for Rows containing Null values in acconts_df """

9

```
# Filter rows where any of the columns have null values
null_rows_accounts_df = accounts_df.filter(
    (accounts_df["account_id"].isNull()) |
    (accounts_df["customer_id"].isNull()) |
    (accounts_df["account_type"].isNull()) |
    (accounts_df["balance"].isNull())
)

# Show the rows containing null values
null_rows_accounts_df.show()
```

```
+----------+-----------+------------+-------+
|account_id|customer_id|account_type|balance|
+----------+-----------+------------+-------+
+----------+-----------+------------+-------+
```

""" Identifying and Removing Duplicates """

11

```
accounts_df = accounts_df.dropDuplicates()
customers_df = customers_df.dropDuplicates()
loan_payments_df = loan_payments_df.dropDuplicates()
loans_df = loans_df.dropDuplicates()
transactions_df = transactions_df.dropDuplicates()
```

Data transformation

""" Filter out rows from accounts_df where the balance column is less than 500 and make those changes in other table """

14

```
# Remove rows where the balance in accounts.csv is < 500
accounts_df = accounts_df.filter(accounts_df["balance"] >= 500)
```

15

```
# Get the number of rows in the DataFrame
row_count = accounts_df.count()

print(f"The number of rows in accounts.csv is: {row_count}")
```

```
The number of rows in accounts.csv is: 79
```

Filter customers Table

17

```
# Get distinct customer_ids associated with accounts that have a balance >= 500
filtered_customer_ids = accounts_df.select("customer_id").distinct()
```

18

```
# Join with filtered_customer_ids to keep only relevant customers
customers_df = customers_df.join(filtered_customer_ids, on="customer_id", how="inner")
```

Filter loans Table

20

```
# Join loans_df with filtered_customer_ids to keep only relevant loans
loans_df = loans_df.join(filtered_customer_ids, on="customer_id", how="inner")
```

Filter payment_loans Table

22

```
# Get distinct loan_ids from filtered loans_df
filtered_loan_ids = loans_df.select("loan_id").distinct()

# Filter payment_loans_df using the filtered loan_ids
loan_payments_df = loan_payments_df.join(filtered_loan_ids, on="loan_id", how="inner")
```

Filter transactions Table

24

```
# Get distinct account_ids from filtered accounts_df
filtered_account_ids = accounts_df.select("account_id").distinct()

# Filter transactions_df using the filtered account_ids
transactions_df = transactions_df.join(filtered_account_ids, on="account_id", how="inner")
```

""" Count rows after data cleaning """

26

```
# Get the number of rows in the DataFrame
row_count = accounts_df.count()
row_count = customers_df.count()
row_count = loan_payments_df.count()
row_count = loans_df.count()
row_count = transactions_df.count()

# Display the number of rows
print(f"The number of rows in customers.csv is: {row_count}")

print(f"The number of rows in accounts.csv is: {row_count}")

print(f"The number of rows in loan_payments.csv is: {row_count}")

print(f"The number of rows in loans.csv is: {row_count}")

print(f"The number of rows in transactions.csv is: {row_count}")
```

```
The number of rows in customers.csv is: 78
The number of rows in accounts.csv is: 78
The number of rows in loan_payments.csv is: 78
The number of rows in loans.csv is: 78
The number of rows in transactions.csv is: 78
```

Save DataFrames to the Silver Container

28

```
# Define your storage account and container details
storage_account_name = "ncplstorageact"
container_name = "silver"
mount_point = f"/mnt/{container_name}"

# Check if the container is already mounted
if not any(mount.mountPoint == mount_point for mount in dbutils.fs.mounts()):
    dbutils.fs.mount(
        source=f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net",
        mount_point=mount_point,
        extra_configs={
            f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net":
dbutils.secrets.get('testScope', 'secret-ncplstorageact')
        }
    )
```

29

```
# Mount the Silver container
mount_point = "/mnt/silver"
```

30