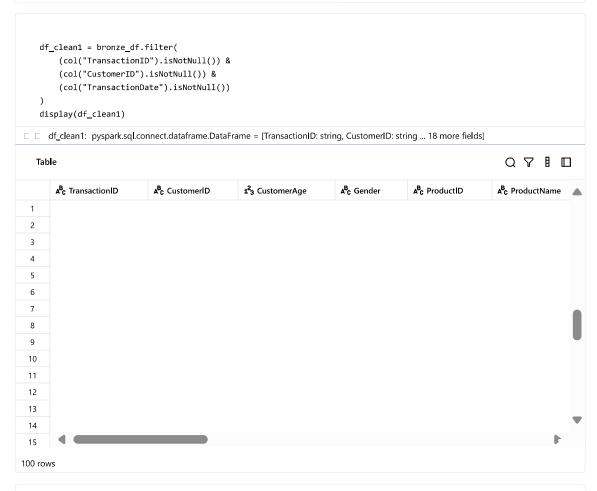


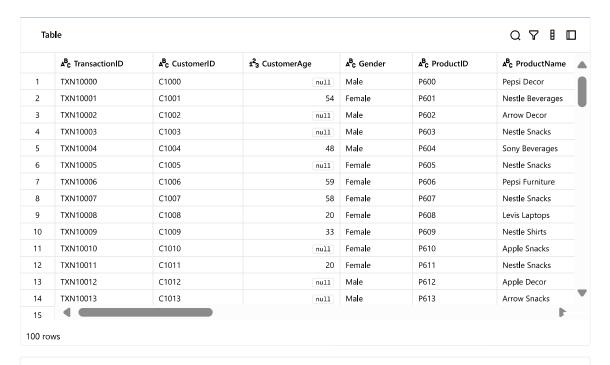
□ □ bronze_df: pyspark.sql.connect.dataframe.DataFrame = [TransactionID: string, CustomerID: string ... 18 more fields]

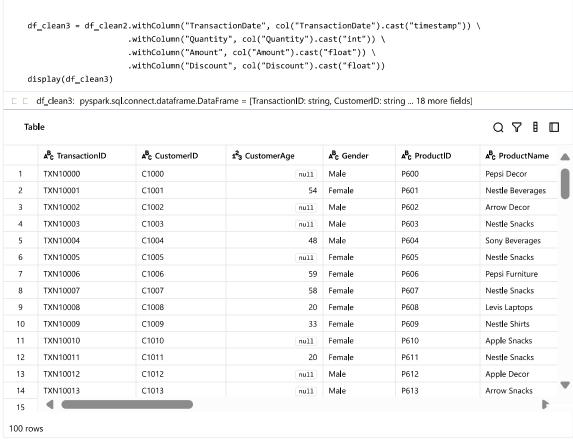
Table

Q7 I D

^B_C CustomerID ^Bc Gender A^B_C ProductID △Bc ProductName A^B_C TransactionID 123 CustomerAge TXN10000 C1000 null Male Pepsi Decor 1 2 TXN10001 C1001 54 Female P601 Nestle Beverages TXN10002 C1002 P602 Arrow Decor 3 null Male C1003 4 TXN10003 P603 Nestle Snacks null Male TXN10004 C1004 Male Sony Beverages 5 48 P604 TXN10005 C1005 null Female P605 Nestle Snacks 7 TXN10006 C1006 Female P606 Pepsi Furniture 59 TXN10007 C1007 Female P607 Nestle Snacks 8 58 9 TXN10008 C1008 20 Female P608 Levis Laptops 10 TXN10009 C1009 33 Female P609 Nestle Shirts C1010 11 TXN10010 Female P610 Apple Snacks null TXN10011 C1011 Female P611 Nestle Snacks 12 20 13 TXN10012 C1012 null Male P612 Apple Decor 14 TXN10013 C1013 null Male P613 Arrow Snacks 15 100 rows







```
# Step 5: Filter out invalid quantity, negative values, excessive discounts
    df_clean4 = df_clean3.filter(
         (col("Quantity") > 0) &
         (col("Amount") > 0) &
         (col("Discount") <= col("Amount")))</pre>
    display(df_clean4)
🗆 🗖 df_clean4: pyspark.sql.connect.dataframe.DataFrame = [TransactionID: string, CustomerID: string ... 18 more fields]
                                                                                                                   QTID
   Table
        ABC TransactionID
                               ABC CustomerID
                                                     123 CustomerAge
                                                                            A<sup>B</sup>c Gender
                                                                                             ABc ProductID
                                                                                                                 A<sup>B</sup>C ProductName
        TXN10000
                               C1000
                                                                                             P600
                                                                    null
                                                                           Male
                                                                                                                 Pepsi Decor
  2
        TXN10001
                               C1001
                                                                       54
                                                                           Female
                                                                                             P601
                                                                                                                 Nestle Beverages
  3
        TXN10002
                               C1002
                                                                    null
                                                                           Male
                                                                                             P602
                                                                                                                 Arrow Decor
        TXN10003
                               C1003
                                                                                             P603
  4
                                                                    null
                                                                           Male
                                                                                                                 Nestle Snacks
        TXN10004
                               C1004
                                                                       48
                                                                           Male
                                                                                             P604
                                                                                                                 Sony Beverages
  6
        TXN10005
                               C1005
                                                                           Female
                                                                                             P605
                                                                                                                 Nestle Snacks
  7
        TXN10006
                               C1006
                                                                           Female
                                                                                             P606
                                                                                                                 Pepsi Furniture
                                                                       59
  8
        TXN10007
                               C1007
                                                                       58
                                                                           Female
                                                                                             P607
                                                                                                                 Nestle Snacks
        TXN10008
                               C1008
                                                                           Female
                                                                                             P608
                                                                                                                 Levis Laptops
 10
        TXN10009
                               C1009
                                                                       33
                                                                           Female
                                                                                             P609
                                                                                                                 Nestle Shirts
 11
        TXN10010
                               C1010
                                                                           Female
                                                                                             P610
                                                                                                                 Apple Snacks
                                                                    null
 12
        TXN10011
                               C1011
                                                                          Female
                                                                                             P611
                                                                                                                 Nestle Snacks
                                                                       20
        TXN10012
                               C1012
                                                                                             P612
                                                                                                                 Apple Decor
 13
                                                                    null
                                                                           Male
 14
        TXN10013
                               C1013
                                                                    null
                                                                           Male
                                                                                             P613
                                                                                                                 Arrow Snacks
        4
 15
100 rows
    df_clean5 = df_clean4.dropDuplicates(["TransactionID"])
□ df_clean5: pyspark.sql.connect.dataframe.DataFrame = [TransactionID: string, CustomerID: string ... 18 more fields]
    # DBTITLE 1, silver dataset
    # Step 2: Drop critical nulls
    df_clean1 = bronze_df.filter(
         (col("TransactionID").isNotNull()) &
         (col("CustomerID").isNotNull()) &
         (col("TransactionDate").isNotNull())
    )
🗆 🗖 df_clean1: pyspark.sql.connect.dataframe.DataFrame = [TransactionID: string, CustomerID: string ... 18 more fields]
    # DBTITLE 1, silver dataset
    # Step 2: Drop critical nulls
    df_clean1 = bronze_df.filter(
         (col("TransactionID").isNotNull()) &
         (col("CustomerID").isNotNull()) &
         (col("TransactionDate").isNotNull())
    )
🗆 🗖 df_clean1: pyspark.sql.connect.dataframe.DataFrame = [TransactionID: string, CustomerID: string ... 18 more fields]
```

```
# Step 3: Trim and standardize text fields
    df_clean2 = df_clean1.withColumn("PaymentType", upper(trim(col("PaymentType")))) \
                           .withColumn("StoreRegion", upper(trim(col("StoreRegion")))) \
                           .withColumn("DeviceUsed", upper(trim(col("DeviceUsed"))))
🗆 🖒 df_clean2: pyspark.sql.connect.dataframe.DataFrame = [TransactionID: string, CustomerID: string ... 18 more fields]
    # Step 4: Cast types
    df_clean3 = df_clean2.withColumn("TransactionDate", col("TransactionDate").cast("timestamp")) \
                           .withColumn("Quantity", col("Quantity").cast("int")) \
                           .withColumn("Amount", col("Amount").cast("float")) \
                           .withColumn("Discount", col("Discount").cast("float"))
🗆 🗅 df_clean3: pyspark.sql.connect.dataframe.DataFrame = [TransactionID: string, CustomerID: string ... 18 more fields]
    # Step 5: Filter out invalid quantity, negative values, excessive discounts
    df_clean4 = df_clean3.filter(
        (col("Quantity") > 0) &
        (col("Amount") > 0) &
        (col("Discount") <= col("Amount"))</pre>
    )
🗆 🗖 df_clean4: pyspark.sql.connect.dataframe.DataFrame = [Transaction|D: string, Customer|D: string ... 18 more fields]
    # Step 6: Drop duplicates
    df_clean5 = df_clean4.dropDuplicates(["TransactionID"])
□ df_clean5: pyspark.sql.connect.dataframe.DataFrame = [TransactionID: string, CustomerID: string ... 18 more fields]
    # Step 7: Write to Silver
    df_clean5.write.format("parquet").mode("overwrite").save("/mnt/storagename/silver")
    silver_df=spark.read.parquet('/mnt/storagename/silver/')
    display(silver df)
🗅 🗠 silver_df: pyspark.sql.connect.dataframe.DataFrame = [TransactionID: string, CustomerID: string ... 18 more fields]
   Table
```

DBTITLE 1,gold dataset
<pre>silver_df.createOrReplaceTempView('retail_data')</pre>
%sql
select * from retail_data
∟ _sqldf: pyspark.sql.connect.dataframe.DataFrame = [TransactionID: string, CustomerID: string 18 more fields]
Table
ianje
This result is stored as _sqldf and can be used in other Python and SQL cells.
DBTITLE 1,daily revenue by purchase
<pre>%sql select date(TransactionDate),sum(amount) total_revenue,count(distinct TransactionID) total_purchase from</pre>
retail_data retail_data
group by 1
• > SyntaxError: invalid syntax (command-6153654693092673-521328151, line 3) [Trace ID: 00-9a241ec474f562b1b5adc522039d04e3-e454ab60c0e78515-00]
[ace 10. 00-76241ec474 302010360C322033004e3-e4348000C0e70313-00]

```
%sql
    SELECT
        DATE(TransactionDate) AS transaction_date,
        SUM(amount) AS total_revenue,
        {\tt COUNT(DISTINCT\ TransactionID)\ AS\ total\_purchases}
    FROM
        retail_data
    GROUP BY
        DATE(TransactionDate);
🗆 💆 _sqldf: pyspark.sql.connect.dataframe.DataFrame = [transaction_date: date, total_revenue: double ... 1 more field]
   Table
This result is stored as _sqldf and can be used in other Python and SQL cells.
# DBTITLE 1, REVENUE BY PAYMENT TYPE
 %sql
  select sum(amount) total_revenue, PAYMENTTYPE from retail_data
  group by 2
• IndentationError: unexpected indent (command-6153654693092675-4029064299, line 2)
[Trace ID: 00-5f53a740d267cb0aa2d0acebdd3b82b4-94a1e1b80d2051a3-00]
    %sql
    select sum(amount) total_revenue, PAYMENTTYPE from retail_data
     group by PAYMENTTYPE

        ∟ _sqldf: pyspark.sql.connect.dataframe.DataFrame = [total_revenue: double, PAYMENTTYPE: string]

   Table
```

```
This result is stored as _sqldf and can be used in other Python and SQL cells.
 # DBTITLE 1,STORE PERFORMANCE
 %sql
 select sum(amount) as total_revenue, STORELOCATION from retail_data
  group by STORELOCATION
① > SyntaxError: invalid syntax (command-6153654693092678-2663134886, line 3)
[Trace ID: 00-898ec182064b64cefd40b06c2b2b0425-840e0298e07ac5c8-00]
    %sql
    SELECT
        StoreLocation AS store_location,
        SUM(amount) AS total_revenue
    FROM
        retail_data
    GROUP BY
        StoreLocation;
☐ __sqldf: pyspark.sql.connect.dataframe.DataFrame = [store_location: string, total_revenue: double]
   Table
This result is stored as _sqldf and can be used in other Python and SQL cells.
```

```
# DBTITLE 1,LOYALITY LEVEL REVENUE CONTRIBUTION
   # MAGIC %sql
   # MAGIC select sum(amount) total_revenue,CustomerLoyaltyLevel
   # MAGIC from retail_data
   # MAGIC group by 2
   # COMMAND -----
   # DBTITLE 1, PRODUCT CATEGORY SALES
   # MAGIC %sal
   # MAGIC select sum(amount) total_revenue,ProductCategory
   # MAGIC
   # MAGIC from retail_data
   # MAGIC group by 2
# DBTITLE 1,LOYALITY LEVEL REVENUE CONTRIBUTION
 SELECT CustomerLoyaltyLevel, SUM(amount) AS total_revenue
 FROM retail_data
 GROUP BY CustomerLoyaltyLevel;
• IndentationError: unexpected indent (command-6153654693092681-1704461844, line 3)
[Trace ID: 00-4fd7ae13d1cbac3703599a174c2e0b8c-26e706ea27f525f0-00]
   %sql
   SELECT
       SUM(amount) AS total_revenue,
       CustomerLoyaltyLevel
   FROM retail_data
   GROUP BY CustomerLoyaltyLevel
   ORDER BY total_revenue DESC
☐ _sqldf: pyspark.sql.connect.dataframe.DataFrame = [total_revenue: double, CustomerLoyaltyLevel: string]
   Table
i This result is stored as _sqldf and can be used in other Python and SQL cells.
 # DBTITLE 1, PRODUCT CATEGORY SALES
 select sum(amount) total_revenue,ProductCategory
  from retail_data
 group by 2
• IndentationError: unindent does not match any outer indentation level (<string>, line 5)
```

□ □ _sqldf:	pyspark.sql.connect.dataframe.DataFrame = [product_category: string, total_revenue: double]
Table	
This res	ult is stored as _sqldf and can be used in other Python and SQL cells.