Dell Data Engineering Interview Questions with Programs - 2025

1. Streaming Data Pipeline (Real-Time)

```
Use Kafka + Spark Structured Streaming + Delta Lake.

PySpark Example:
from pyspark.sql import SparkSession
from pyspark.sql.functions import expr

spark = SparkSession.builder.appName("StreamProcessor").getOrCreate()
df = spark.readStream.format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "iot_topic") \
    .load()

value_df = df.selectExpr("CAST(value AS STRING)")
parsed_df = value_df.selectExpr("split(value, ',')[0] as sensor_id",
                                "split(value, ',')[1] as temperature")

query = parsed_df.writeStream \
    .format("delta") \
    .option("checkpointLocation", "/delta/checkpoint") \
    .start("/delta/output")
```

2. SCD Type 2 Implementation

```
Track history by closing old records and inserting new ones.

PySpark:
from pyspark.sql.functions import current_date, lit

existing = spark.read.format("delta").load("/delta/customer")
new = spark.read.csv("/new/customer.csv", header=True)

changes = new.join(existing, "customer_id") \
    .filter(new["name"] != existing["name"])

expired = changes.select(existing["*"]).withColumn("is_current", lit(False)) \
    .withColumn("end_date", current_date())

new_versions = changes.select(new["*"]).withColumn("is_current", lit(True)) \
    .withColumn("start_date", current_date()) \
    .withColumn("end_date", lit(None).cast("date"))

final = existing.unionByName(expired).unionByName(new_versions)
final.write.format("delta").mode("overwrite").save("/delta/customer")
```

3. Schema Evolution Handling

```
Use Delta Lake with mergeSchema:
df = spark.read.json("/new/data.json")
df.write.option("mergeSchema", "true").format("delta").mode("append").save("/delta/data")
```

4. Partitioned Table Write (Spark)

```
Partition data to improve query speed:
df = spark.read.csv("sales_data.csv", header=True)
df.write.partitionBy("region", "year").format("parquet").save("/output/sales")
```

5. Optimize Spark Performance

```
Use broadcast joins, caching, repartitioning.

Broadcast Join:
from pyspark.sql.functions import broadcast
large_df = spark.read.parquet("large_table")
small_df = spark.read.csv("small_lookup.csv", header=True)
joined = large_df.join(broadcast(small_df), "id")
```

6. Delta Lake Time Travel

```
View older versions of Delta data:
old_df = spark.read.format("delta").option("versionAsOf", 2).load("/delta/sales")
```

7. Data Deduplication in PySpark

```
Remove duplicates while keeping the latest:
from pyspark.sql.window import Window
from pyspark.sql.functions import row_number

windowSpec = Window.partitionBy("id").orderBy(df["update_ts"].desc())
df = df.withColumn("row_num", row_number().over(windowSpec)).filter("row_num = 1")
```

8. Data Quality Check

```
Ensure data quality before loading:
from pyspark.sql.functions import col

invalid = df.filter(col("email").isNull() | (col("age") < 0))
if invalid.count() > 0:
    raise Exception("Data Quality Issue Detected!")
```

9. SQL Query for Aggregation

```
Top 3 products by sales per region:

SQL:
SELECT region, product_id, total_sales
FROM (
  SELECT region, product_id, SUM(sales) as total_sales,
         RANK() OVER (PARTITION BY region ORDER BY SUM(sales) DESC) as rnk
  FROM sales_data
  GROUP BY region, product_id
) t
WHERE rnk <= 3;
```

10. PySpark Join Types

```
Show all types of joins:

df1 = spark.createDataFrame([(1, "A"), (2, "B")], ["id", "name"])
df2 = spark.createDataFrame([(1, "X"), (3, "Y")], ["id", "value"])
```

```
df1.join(df2, "id", "inner").show()
df1.join(df2, "id", "left").show()
df1.join(df2, "id", "right").show()
df1.join(df2, "id", "outer").show()
```