Databricks Delta Schema Issues with Mitigations

#	Schema Issue	Description	Where It Typically Arises	Delta Layer(s) Most Affected	Mitigation Strategies
1	Schema Drift	Unexpected changes in incoming data structure (new, missing, renamed columns).	Ingestion (Auto Loader, streaming reads)	Bronze	Use mergeSchema=true, or store raw JSON/text for later parsing.
2	Schema Evolution	Automatic incorporation of schema changes into Delta tables.	Writes to Delta tables with mergeSchema	Bronze, Silver	Enable mergeSchema in writes; review schema history regularly.
3	Data Type Conflicts	Incoming data columns have types that don't match the table schema.	Ingestion, append writes to Delta	Bronze, Silver	Cast fields to consistent types; validate schema before writes.
4	Nullability Mismatches	Changes in whether a column allows nulls, causing constraint violations.	Overwrites, schema enforcement	Silver, Gold	Use ALTER TABLE to adjust nullability; standardize null handling in ETL.
5	Column Reordering	Columns arrive in unexpected order, potentially breaking mappings.	Batch ingestion with schema inference	Bronze	Define explicit schemas; map columns by name rather than position.
6	Nested Field Changes	New fields or type changes inside nested structs or JSON columns.	Ingestion of semi-structured data	Bronze, Silver	Store raw JSON; parse in Silver with explicit schemas and controlled evolution.
7	Column Dropping	Expected columns disappear from incoming data, resulting in nulls	Ingestion pipelines	Bronze, Silver	Fill missing fields with defaults or nulls; alert on schema gaps.

		or failures.			
8	Column Renaming	Columns are renamed upstream without notice, breaking queries and transformations.	Ingestion and transformations	Bronze, Silver	Use mapping tables; rename columns explicitly in ETL jobs.
9	Schema Inference Variability	Automatic inference guesses inconsistent types across files.	Auto Loader, .option("inferSchema", "true")	Bronze	Always define explicit schemas; avoid schema inference in production.
10	Backward Incompatible Changes	Schema changes that can't be merged automatically and require manual fixes.	Table evolution (dropping/retyping columns)	Silver, Gold	Use time travel to restore previous versions; plan and validate schema changes.

Schema Issues and Mitigation Strategies by Delta Layer

#	Schema Issue	Delta Layer(s)	Mitigation Strategy
1	**Schema Drift**	Bronze	Use mergeSchema to accept new columns: python df = (spark.readStream.format("cloudFiles").option("cloudFiles.format","json").option("mergeSchema","true").load("/mnt/raw/")) Store raw JSON: python df = spark.readStream.format("cloudFiles").option("cloudFiles.format","text").load("/mnt/raw/")
2	**Schema Evolution**	Bronze, Silver	Enable mergeSchema: python df.write.option("mergeSchema","true").format("delta").mode("append").save("/mnt/delta/bronze") Track schema changes: sql DESCRIBE HISTORY delta.\`/mnt/delta/bronze\`
3	**Data Type Conflicts**	Bronze, Silver	Cast types before writing: python df = df.withColumn("id", col("id").cast("string")) Store raw text and parse in Silver:

			python df = spark.readStream.format("cloudFiles").option("cloudFiles.format","text").load("/mnt/raw/")
4	**Nullability Mismatches**	Silver, Gold	Adjust nullability: sql ALTER TABLE silver_table ALTER COLUMN user_id DROP NOT NULL
5	**Column Reordering**	Bronze	Define schemas explicitly: python schema = StructType([...]) df = spark.read.schema(schema).json("/mnt/raw/")
6	**Nested Field Changes**	Bronze, Silver	Store raw JSON in Bronze: python df = spark.readStream.format("cloudFiles").option("cloudFiles.format","text").load("/mnt/raw/") Parse in Silver: python parsed_df = df.withColumn("data", from_json(col("value"), schema))
7	**Column Dropping**	Bronze, Silver	Fill defaults in Silver: python df = df.withColumn("event_type", coalesce(col("event_type"), lit("unknown"))) Alert on missing columns: python expected = {"user_id","event_type"} actual = set(df.columns) if missing := expected - actual: print("Missing:", missing)
8	**Column Renaming**	Bronze, Silver	Map renamed fields: python df = df.withColumnRenamed("old_name","new_name")
9	**Schema Inference Variability**	Bronze	Avoid inference: python schema = StructType([...]) df = spark.read.schema(schema).json("/mnt/raw/")
10	**Backward Incompatible Changes**	Silver, Gold	Time travel recovery: python df = spark.read.format("delta").option("versionAsOf",3).load("/mnt/delta/silver") Test schema changes: sql DESCRIBE HISTORY delta.`/mnt/delta/silver`