Healthcare Claims Fraud Detection

Capstone Project – Databricks with Delta Live Tables

Presented by
Bianca Joyce Quinia

Problem Statement

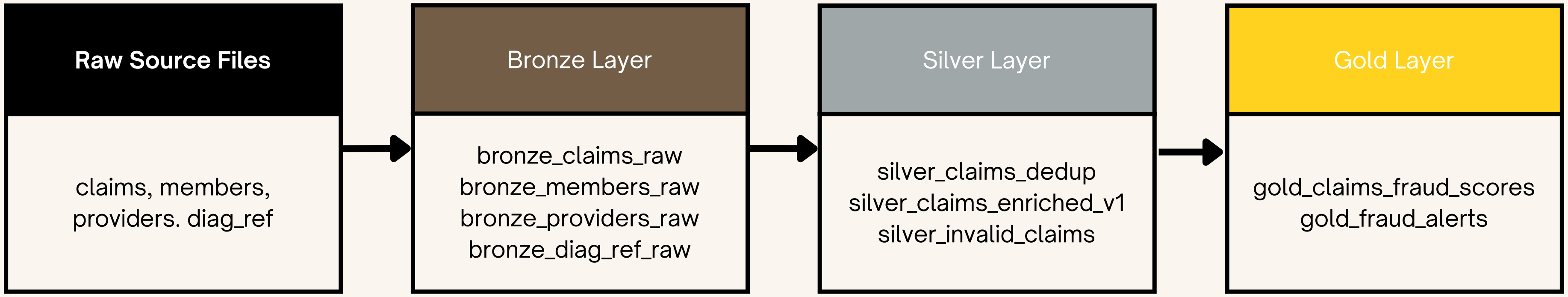
- Traditional ETL lacks scalability, governance, and real-time insights.
- Objective: Build a scalable, governed pipeline to detect fraud in claims data.

Solution Overview

- Built using Databricks + Delta Live Tables
- Applied Medallion Architecture (Bronze → Silver → Gold)
- Ingested batch + streaming data
- Enforced governance (quality checks, history, restore, z-order)
- Produced fraud scores and alerts

Medallion Architecture

Bronze keeps raw fidelity for auditing.
Silver provides clean, consistent, validated data.
Gold delivers business-ready insights (fraud scores & alerts).



Bronze Layer (Raw Ingestion)

- Ingested raw files (CSV, JSON) into Bronze tables:
    - **bronze_claims_raw**
    - **bronze_members_raw**
    - **bronze_providers_raw**
    - **bronze_diag_ref_raw**

Bronze Layer: This layer ingests raw, unprocessed data, including claims, members, providers, and diagnosis references, preserving its original fidelity for auditing purposes.

Silver Layer (Cleaning & Enrichment)

Silver Layer: Here, data is cleaned, validated, and enriched. Key steps include:

- Deduplication: Claims are deduplicated based on ClaimID, keeping the latest record by ingest_ts.
- Enrichment: Claims data is joined with member, provider, and diagnosis reference information.
- Validation: Flags such as member_exists, provider_exists, diagnosis_high_risk, and is_valid are added to claims.

Gold Layer (Fraud Scoring & Alerts)

- **gold_claims_fraud_scores**: Fraud score = score_amount + score_diag + score_provider
- Risk buckets: High / Medium / Low
- **gold_fraud_alerts**: high-risk claims for monitoring.

Gold Layer: This final layer delivers business-ready data, producing fraud scores and alerts.

📈 Fraud Scoring Logic

The pipeline calculates a multi-signal
fraud_score for each claim in the gold_claims_fraud_scores table. This score is based on a combination of factors:

- score_amount: A score of 0.6 is added if the claim's Amount exceeds a $1,000 threshold.
- score_diag: A score of 0.3 is added if the diagnosis_description contains keywords like "CANCER," "MALIGNANT," or "CRITICAL," or if the Diagnosis_Code is on a fallback high-risk list ("D123", "X999").
- score_provider: A score of 0.4 is added if the provider is flagged as inactive.

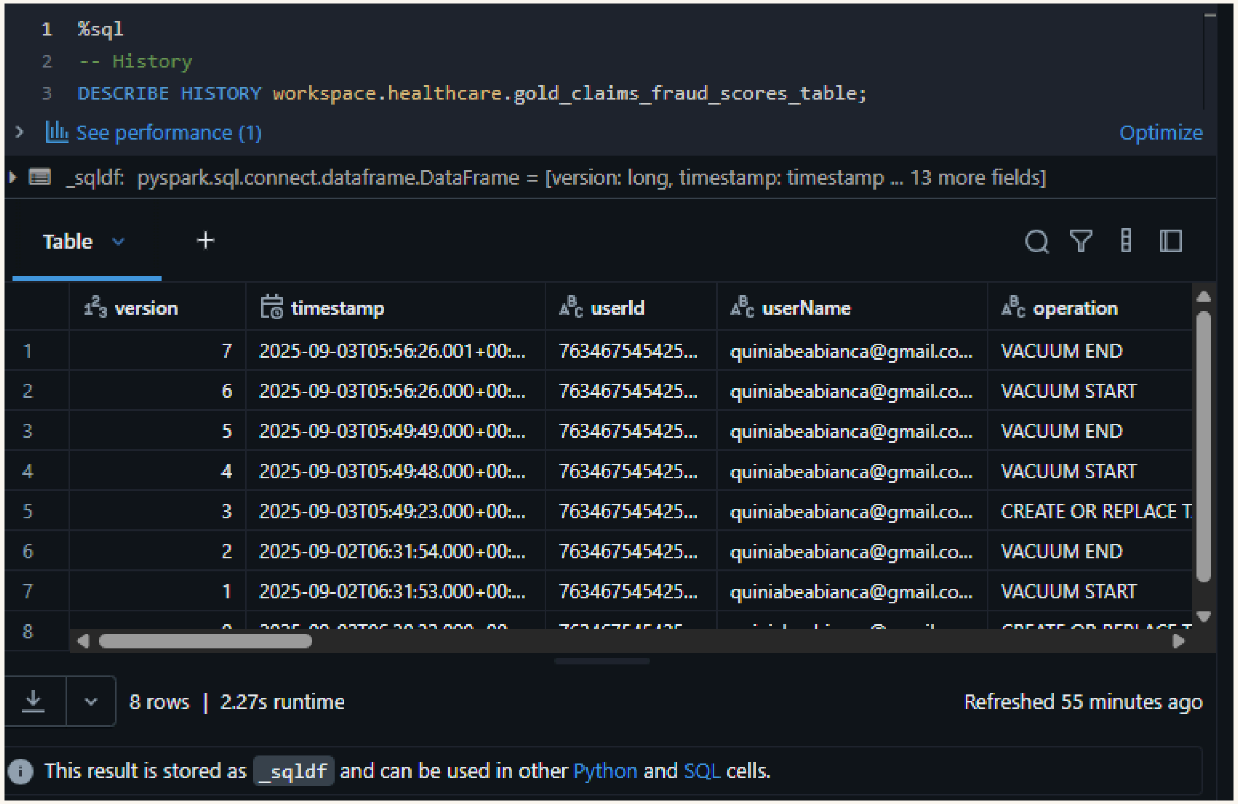
The total : fraud_score is the sum of these individual scores. Claims are then assigned to a risk_bucket (High, Medium, or Low) based on the total score.

- High Risk: fraud_score >= 0.7
- Medium Risk: fraud_score >= 0.3
- Low Risk: fraud_score < 0.3

A separate table, gold_fraud_alerts, is generated to capture and flag claims in the "high" risk bucket for immediate monitoring.

Governance & Optimization

- Data Quality: dlt.expect and dlt.expect_or_drop are used to define quality expectations for tables, ensuring data integrity
- Auditability: The DESCRIBE HISTORY command is used on Delta tables to view the history of changes, providing a complete audit trail.



```
%sql
-- History
DESCRIBE HISTORY workspace.healthcare.gold_claims_fraud_scores_table;
```

See performance (1) | Optimize

_sqldf: pyspark.sql.connect.dataframe.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

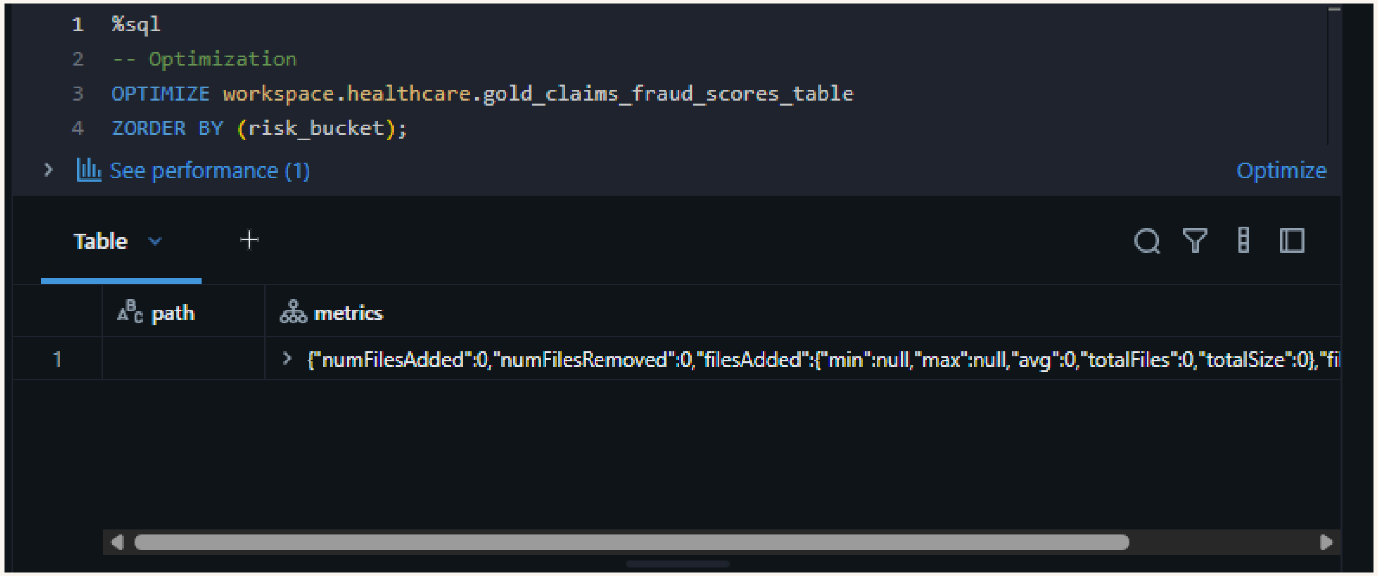
	version	timestamp	userId	userName	operation
1	7	2025-09-03T05:56:26.001+00:...	763467545425...	quiniabeabianca@gmail.co...	VACUUM END
2	6	2025-09-03T05:56:26.000+00:...	763467545425...	quiniabeabianca@gmail.co...	VACUUM START
3	5	2025-09-03T05:49:49.000+00:...	763467545425...	quiniabeabianca@gmail.co...	VACUUM END
4	4	2025-09-03T05:49:48.000+00:...	763467545425...	quiniabeabianca@gmail.co...	VACUUM START
5	3	2025-09-03T05:49:23.000+00:...	763467545425...	quiniabeabianca@gmail.co...	CREATE OR REPLACE T
6	2	2025-09-02T06:31:54.000+00:...	763467545425...	quiniabeabianca@gmail.co...	VACUUM END
7	1	2025-09-02T06:31:53.000+00:...	763467545425...	quiniabeabianca@gmail.co...	VACUUM START
8	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

8 rows | 2.27s runtime — Refreshed 55 minutes ago

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

Governance & Optimization

- Performance: The OPTIMIZE … ZORDER command is used to co-locate related data by risk_bucket, which improves query performance for the gold_claims_fraud_scores

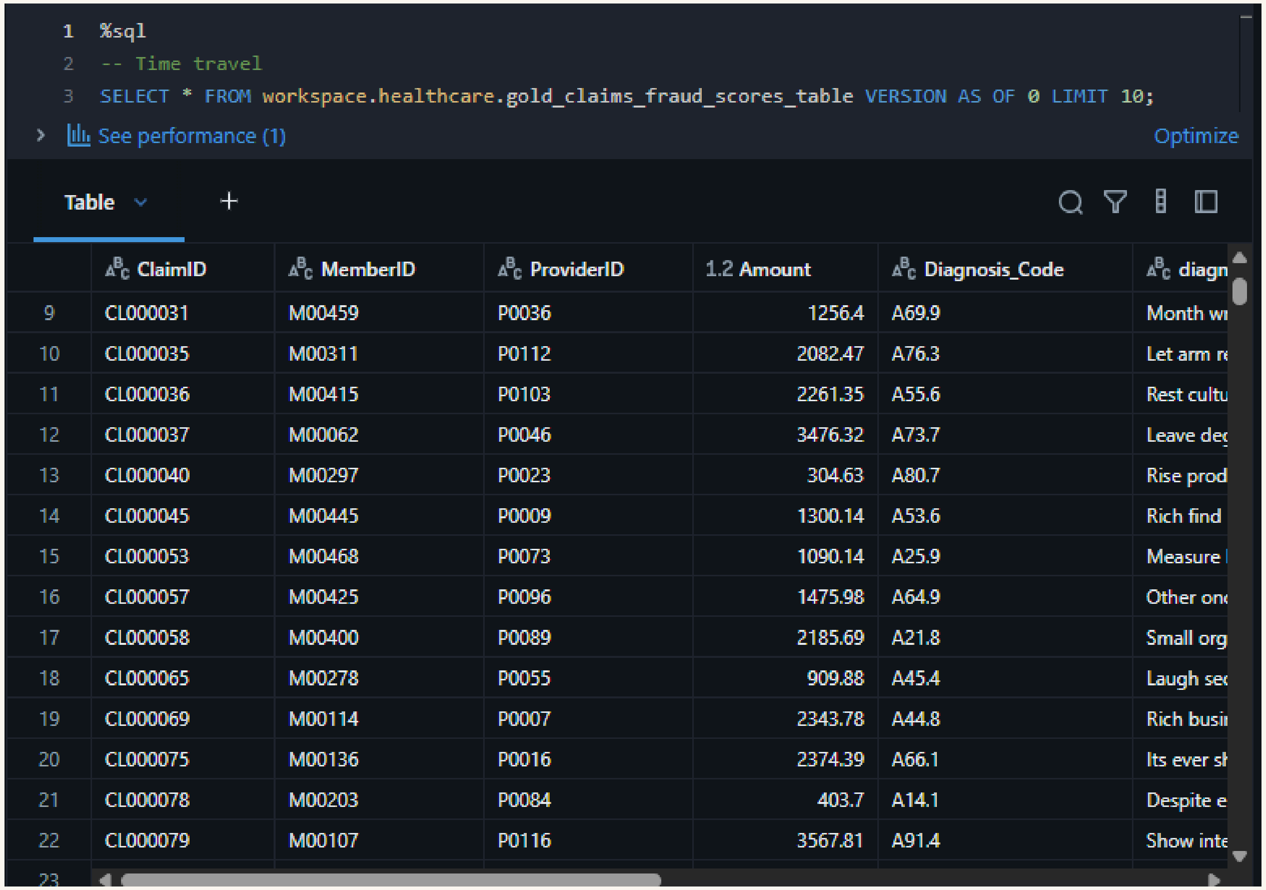


```
%sql
-- Optimization
OPTIMIZE workspace.healthcare.gold_claims_fraud_scores_table
ZORDER BY (risk_bucket);
```

```
   path   metrics
1         {"numFilesAdded":0,"numFilesRemoved":0,"filesAdded":{"min":null,"max":null,"avg":0,"totalFiles":0,"totalSize":0},"fi
```

Demo Results

- This query would retrieve the state of the gold_claims_fraud_scores table from its initial creation, which is useful for auditing and ensuring data lineage. The ability to restore a table to an earlier version is a critical part of the pipeline's governance and recovery capabilities.



```
%sql
-- Time travel
SELECT * FROM workspace.healthcare.gold_claims_fraud_scores_table VERSION AS OF 0 LIMIT 10;
```

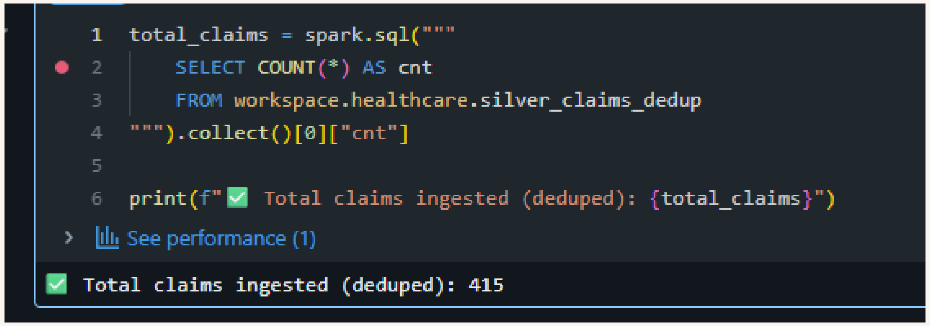
See performance (1)

Optimize

Table

	ClaimID	MemberID	ProviderID	1.2 Amount	Diagnosis_Code	diagn
9	CL000031	M00459	P0036	1256.4	A69.9	Month wi
10	CL000035	M00311	P0112	2082.47	A76.3	Let arm re
11	CL000036	M00415	P0103	2261.35	A55.6	Rest cultu
12	CL000037	M00062	P0046	3476.32	A73.7	Leave deg
13	CL000040	M00297	P0023	304.63	A80.7	Rise prod
14	CL000045	M00445	P0009	1300.14	A53.6	Rich find
15	CL000053	M00468	P0073	1090.14	A25.9	Measure
16	CL000057	M00425	P0096	1475.98	A64.9	Other onc
17	CL000058	M00400	P0089	2185.69	A21.8	Small org
18	CL000065	M00278	P0055	909.88	A45.4	Laugh sec
19	CL000069	M00114	P0007	2343.78	A44.8	Rich busi
20	CL000075	M00136	P0016	2374.39	A66.1	Its ever sh
21	CL000078	M00203	P0084	403.7	A14.1	Despite e
22	CL000079	M00107	P0116	3567.81	A91.4	Show inte
23						

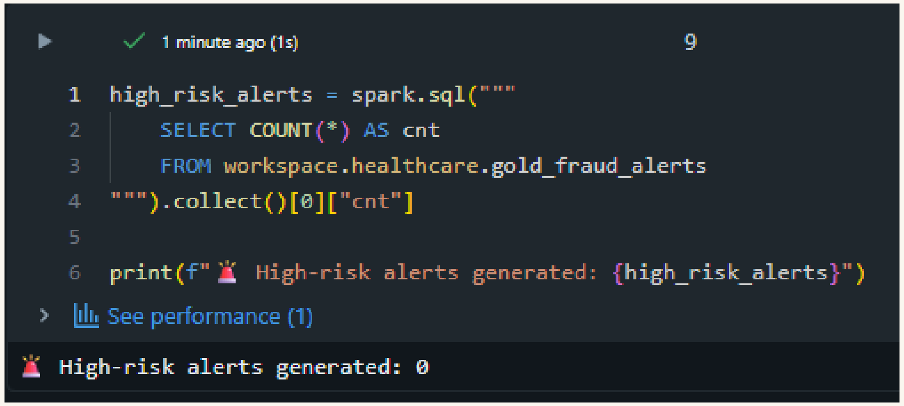
PIPELINE VALIDATION TESTS



```
total_claims = spark.sql("""
    SELECT COUNT(*) AS cnt
    FROM workspace.healthcare.silver_claims_dedup
""").collect()[0]["cnt"]

print(f"✅ Total claims ingested (deduped): {total_claims}")
```

See performance (1)

```
✅ Total claims ingested (deduped): 415
```

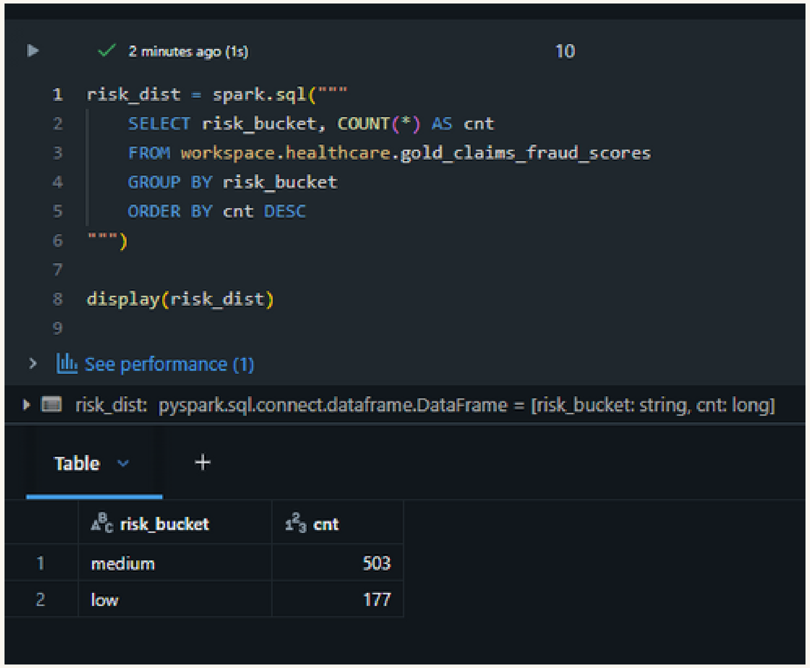


1 minute ago (1s) 9

```
high_risk_alerts = spark.sql("""
    SELECT COUNT(*) AS cnt
    FROM workspace.healthcare.gold_fraud_alerts
""").collect()[0]["cnt"]

print(f"🚨 High-risk alerts generated: {high_risk_alerts}")
```

See performance (1)

```
🚨 High-risk alerts generated: 0
```



2 minutes ago (1s) 10

```
risk_dist = spark.sql("""
    SELECT risk_bucket, COUNT(*) AS cnt
    FROM workspace.healthcare.gold_claims_fraud_scores
    GROUP BY risk_bucket
    ORDER BY cnt DESC
""")

display(risk_dist)
```

See performance (1)

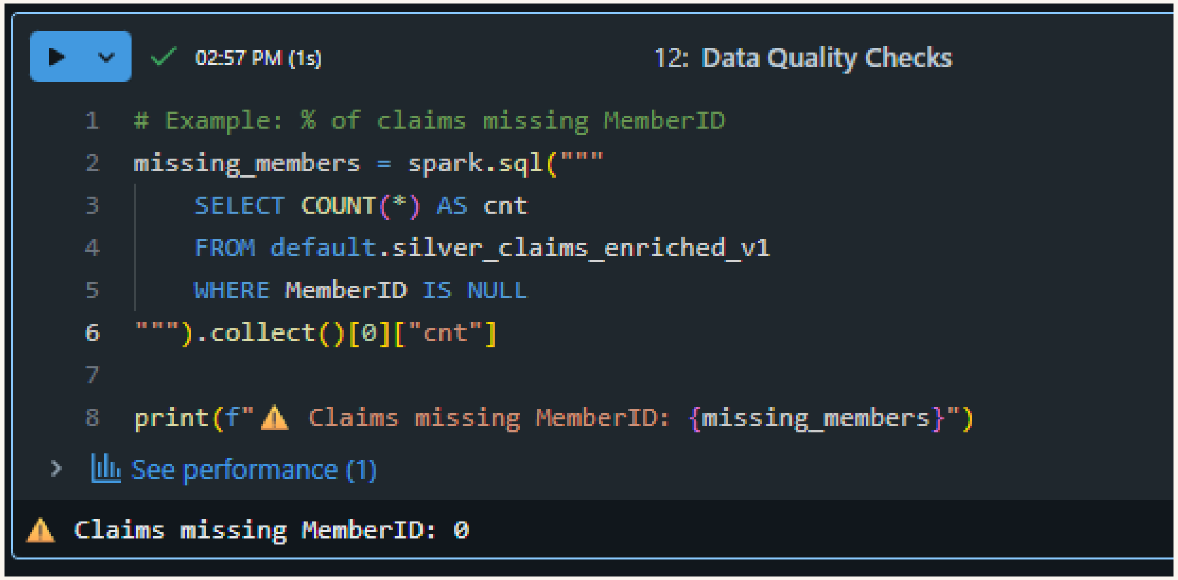
```
risk_dist: pyspark.sql.connect.dataframe.DataFrame = [risk_bucket: string, cnt: long]
```

Table

	risk_bucket	cnt
1	medium	503
2	low	177

There are no fraud alerts since there are no high risk count.

DATA QUALITY CHECKS

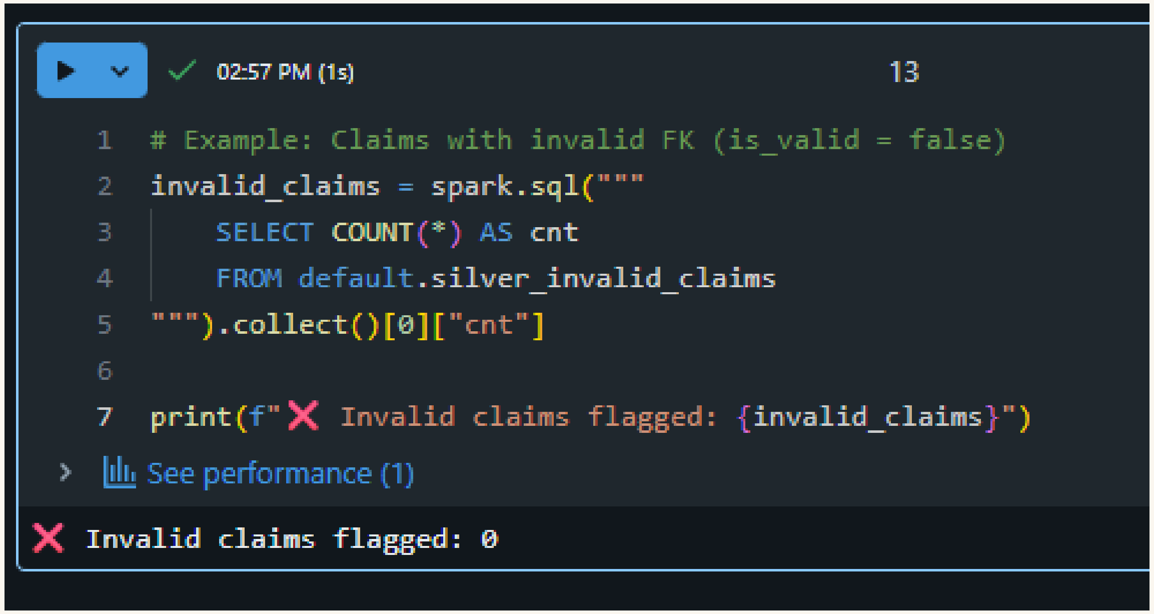


```
# Example: % of claims missing MemberID
missing_members = spark.sql("""
    SELECT COUNT(*) AS cnt
    FROM default.silver_claims_enriched_v1
    WHERE MemberID IS NULL
""").collect()[0]["cnt"]

print(f"⚠ Claims missing MemberID: {missing_members}")
```

```
⚠ Claims missing MemberID: 0
```

- In the silver_claims_enriched_v1 table, all rows have a valid MemberID.
- There are no claims missing a MemberID, so every claim could be joined back to a member record from bronze_members_raw.

DATA QUALITY CHECKS

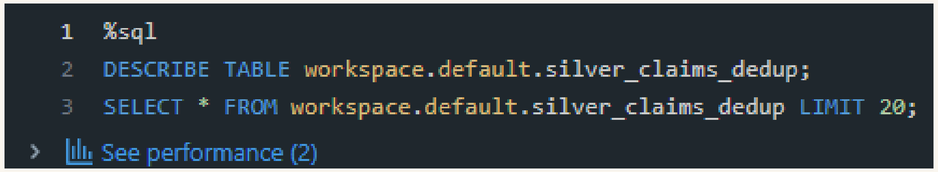


```
# Example: Claims with invalid FK (is_valid = false)
invalid_claims = spark.sql("""
    SELECT COUNT(*) AS cnt
    FROM default.silver_invalid_claims
""").collect()[0]["cnt"]

print(f"❌ Invalid claims flagged: {invalid_claims}")
```

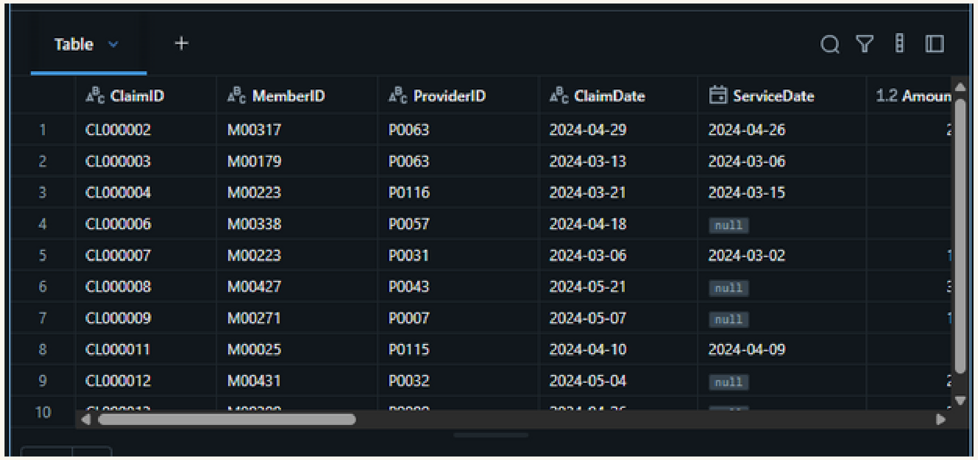
```
❌ Invalid claims flagged: 0
```

- The validation check for invalid claims in the silver_invalid_claims table returned a count of zero. This indicates that no claims were flagged as invalid, suggesting that the dataset maintains high data integrity with respect to the defined validation rules. Consequently, there are no immediate concerns regarding the quality or validity of the claims data.

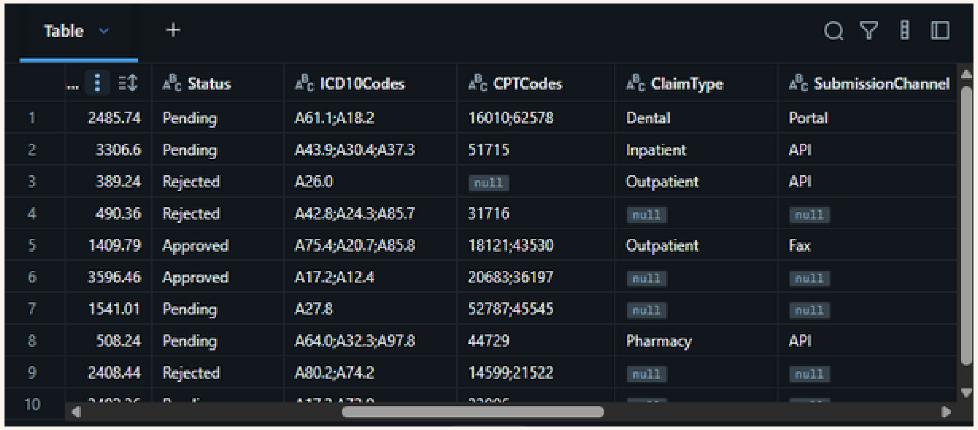
silver_claims_dedup



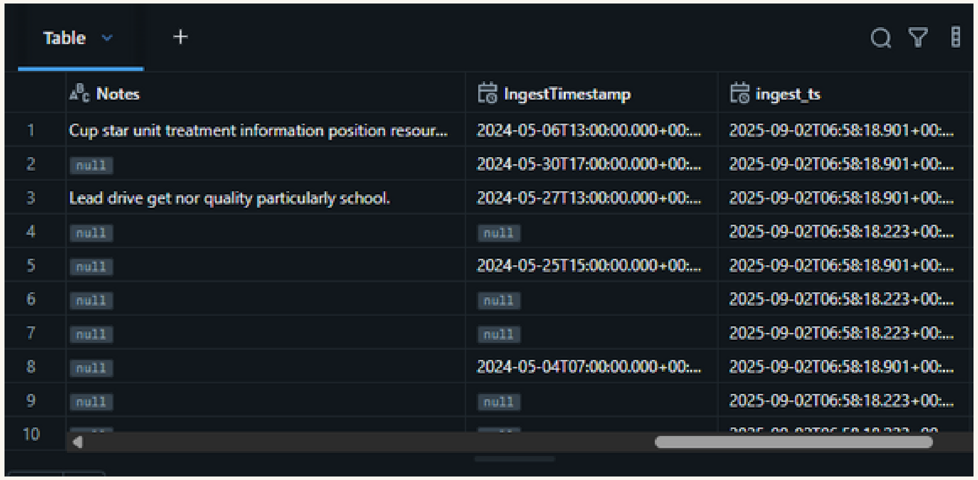
```
%sql
DESCRIBE TABLE workspace.default.silver_claims_dedup;
SELECT * FROM workspace.default.silver_claims_dedup LIMIT 20;
```



	ClaimID	MemberID	ProviderID	ClaimDate	ServiceDate	1.2 Amoun
1	CL000002	M00317	P0063	2024-04-29	2024-04-26	2
2	CL000003	M00179	P0063	2024-03-13	2024-03-06	
3	CL000004	M00223	P0116	2024-03-21	2024-03-15	
4	CL000006	M00338	P0057	2024-04-18	null	
5	CL000007	M00223	P0031	2024-03-06	2024-03-02	1
6	CL000008	M00427	P0043	2024-05-21	null	3
7	CL000009	M00271	P0007	2024-05-07	null	1
8	CL000011	M00025	P0115	2024-04-10	2024-04-09	
9	CL000012	M00431	P0032	2024-05-04	null	2
10	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	



	...	Status	ICD10Codes	CPTCodes	ClaimType	SubmissionChannel
1	2485.74	Pending	A61.1;A18.2	16010;62578	Dental	Portal
2	3306.6	Pending	A43.9;A30.4;A37.3	51715	Inpatient	API
3	389.24	Rejected	A26.0	null	Outpatient	API
4	490.36	Rejected	A42.8;A24.3;A85.7	31716	null	null
5	1409.79	Approved	A75.4;A20.7;A85.8	18121;43530	Outpatient	Fax
6	3596.46	Approved	A17.2;A12.4	20683;36197	null	null
7	1541.01	Pending	A27.8	52787;45545	null	null
8	508.24	Pending	A64.0;A32.3;A97.8	44729	Pharmacy	API
9	2408.44	Rejected	A80.2;A74.2	14599;21522	null	null
10	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

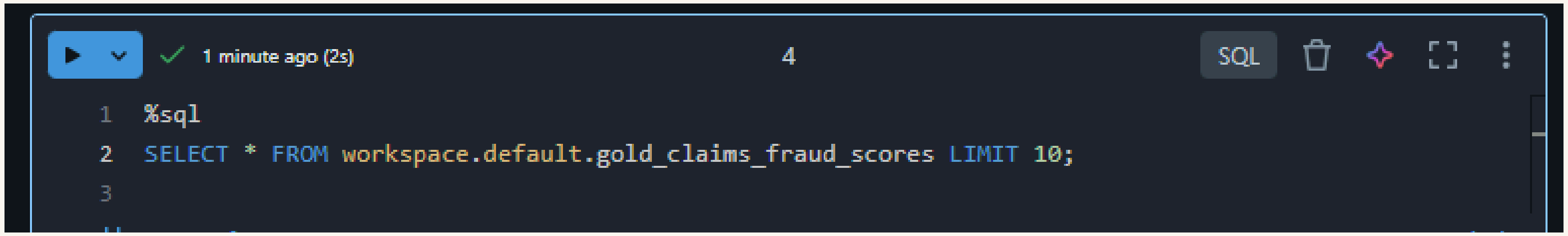


	Notes	IngestTimestamp	ingest_ts
1	Cup star unit treatment information position resour...	2024-05-06T13:00:00.000+00:...	2025-09-02T06:58:18.901+00:...
2	null	2024-05-30T17:00:00.000+00:...	2025-09-02T06:58:18.901+00:...
3	Lead drive get nor quality particularly school.	2024-05-27T13:00:00.000+00:...	2025-09-02T06:58:18.901+00:...
4	null	null	2025-09-02T06:58:18.223+00:...
5	null	2024-05-25T15:00:00.000+00:...	2025-09-02T06:58:18.901+00:...
6	null	null	2025-09-02T06:58:18.223+00:...
7	null	null	2025-09-02T06:58:18.223+00:...
8	null	2024-05-04T07:00:00.000+00:...	2025-09-02T06:58:18.901+00:...
9	null	null	2025-09-02T06:58:18.223+00:...
10	[illegible]	[illegible]	[illegible]

Reads:

- Reads bronze_claims_raw.
- Deduplicates on claim_id (keeps latest by ingest_ts).
- Ensures there's one row per claim.

gold_claims_fraud_scores

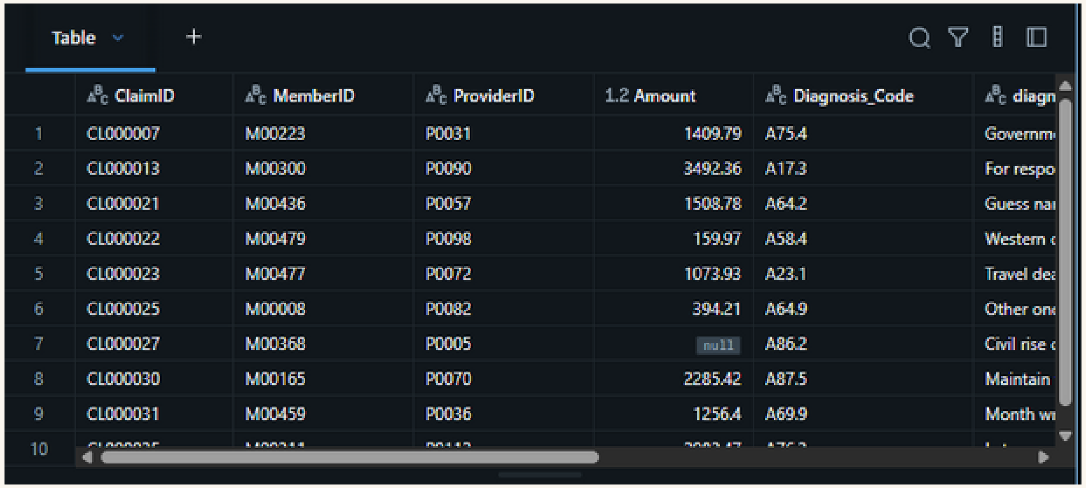


1 minute ago (2s) 4 SQL

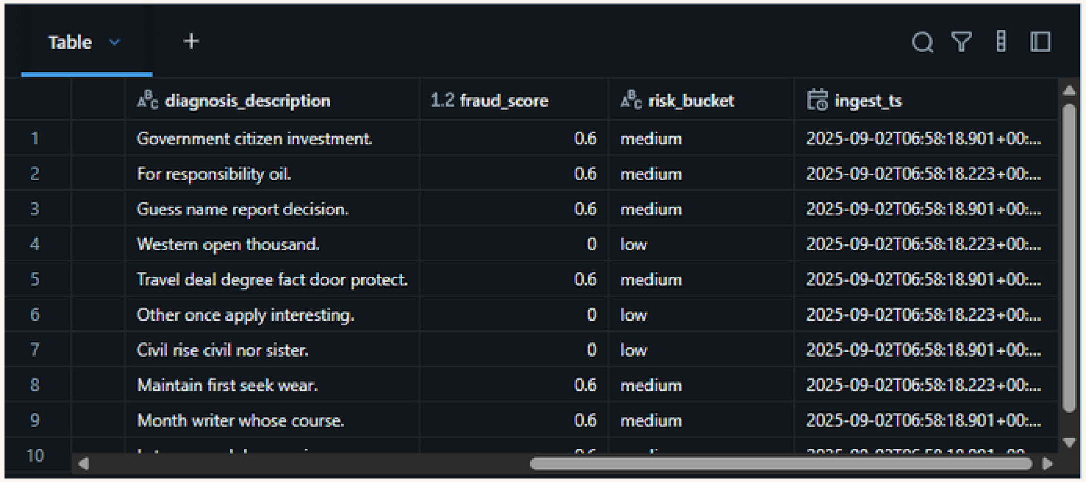
```
%sql
SELECT * FROM workspace.default.gold_claims_fraud_scores LIMIT 10;

```

Multi-signal fraud scoring:

- score_amount: credit for high claim amounts
- score_diag: credit for high-risk diagnosis codes
- score_provider: credit if provider_flagged is True (if provider metadata includes it)



	ClaimID	MemberID	ProviderID	Amount	Diagnosis_Code	diagn
1	CL000007	M00223	P0031	1409.79	A75.4	Governm
2	CL000013	M00300	P0090	3492.36	A17.3	For respo
3	CL000021	M00436	P0057	1508.78	A64.2	Guess nar
4	CL000022	M00479	P0098	159.97	A58.4	Western c
5	CL000023	M00477	P0072	1073.93	A23.1	Travel dea
6	CL000025	M00008	P0082	394.21	A64.9	Other onc
7	CL000027	M00368	P0005	null	A86.2	Civil rise c
8	CL000030	M00165	P0070	2285.42	A87.5	Maintain
9	CL000031	M00459	P0036	1256.4	A69.9	Month wr
10	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]



	diagnosis_description	fraud_score	risk_bucket	ingest_ts
1	Government citizen investment.	0.6	medium	2025-09-02T06:58:18.901+00:...
2	For responsibility oil.	0.6	medium	2025-09-02T06:58:18.223+00:...
3	Guess name report decision.	0.6	medium	2025-09-02T06:58:18.901+00:...
4	Western open thousand.	0	low	2025-09-02T06:58:18.223+00:...
5	Travel deal degree fact door protect.	0.6	medium	2025-09-02T06:58:18.901+00:...
6	Other once apply interesting.	0	low	2025-09-02T06:58:18.223+00:...
7	Civil rise civil nor sister.	0	low	2025-09-02T06:58:18.901+00:...
8	Maintain first seek wear.	0.6	medium	2025-09-02T06:58:18.223+00:...
9	Month writer whose course.	0.6	medium	2025-09-02T06:58:18.901+00:...
10	[illegible]	[illegible]	[illegible]	[illegible]

- **score_amount** = 0.6 if claim_amount > threshold
- **score_diag** = 0.3 if diagnosis_high_risk
- **score_provider** = 0.4 if provider_flagged
- **fraud_score** = score_amount + score_diag + score_provider
- **risk_bucket** = high / medium / low
  - high: If fraud_score is greater than or equal to 0.7.
  - medium: If fraud_score is greater than or equal to 0.3.
  - low: If fraud_score is less than 0.3

Conclusion

- Delivered end-to-end pipeline with:
  - Medallion Architecture (Bronze, Silver, Gold), which ensures raw data fidelity for auditing, provides clean and consistent data, and delivers business-ready insights, respectively. The pipeline effectively handled both batch and streaming data ingestion.
  - Data Governance: The pipeline incorporated crucial governance features like data quality checks (**dlt.expect**, **dlt.expect_or_drop**), auditability via data history, and the ability to restore to previous versions.
  - Fraud Detection: A multi-signal scoring model was developed to calculate a **fraud_score** based on factors such as claim amount, diagnosis codes, and provider status. This enabled the categorization of claims into High, Medium, and Low risk buckets, with high-risk claims flagged for alerts
- Future work: Integrate ML models, external alerting systems.