ETL with Databricks
Delta Live Tables
What’s the problem with Data Engineering?
We know data is critical to business outcomes
But there is complexity in data engineering....

Data Sources

- Streaming Sources
- Cloud Object Stores
- SaaS Applications
- NoSQL
- Relational Databases
- On-premises systems

Unstructured

- Qlik
- Fivetran

Semi-structured

- Rivery

Structured

- AWS Glue
- AWS EMR

Cloud Data Lake

Data sharing

- Azure Data Factory
- Azure Synapse
- AWS Glue
- AWS EMR
- Code Generated

Business Insights

- Machine Learning

Analytics

- Streaming Analytics
- Azure Synapse
- Code Generated

Databricks Inc. — All rights reserved
How does Databricks Help?
Databricks Lakehouse Platform is the foundation for Data Engineering
Declarative Pipelines for ExtractLoad-Transform

- script defining a DAG of ingestion and downstream transform

Ingestion (EL) → Downstream (T)

- kafka
- Kinesis
- CSV, JSON, TXT...
- Data Lake

Streaming Tables++

- Materialized Views++

- Streaming Analytics
- BI & Reporting
- Data Science & ML
Key Differentiators:
Databases meet Software Engineering and Systems
Ingestion

- Incrementally and efficiently process new data files as they arrive in cloud storage using Auto Loader
- Automatically infer schema of incoming files or superimpose what you know with Schema Hints
- Automatic schema evolution
- Rescue data column – never lose data again

Simple SQL syntax for streaming ingestion

```sql
CREATE STREAMING LIVE TABLE sales_orders_raw
COMMENT "The raw sales orders, ingested from /databricks-datasets." TBLPROPERTIES ("myCompanyPipeline.quality" = "bronze") AS
SELECT * FROM cloud_files
("/databricks-datasets/retail-org/sales_orders/", "json", map("cloudFiles.inferColumnType", "true"))
```

Schema Evolution: JSON, CSV, AVRO, PARQUET
Change data capture (CDC)

- Stream change records (inserts, updates, deletes) from any data source supported by DBR, cloud storage, or DBFS
- Simple, declarative “APPLY CHANGES INTO” API for SQL or Python
- Handles out-of-order events
- Partial Updates
- SCD2 support
Data Engineers don’t just code: collaborate, version, test, validate, monitor
Data quality validation and monitoring

- Define data quality and integrity controls within the pipeline with **data expectations**
- Address data quality errors with **flexible policies**: fail, drop, alert, quarantine(future)
- All data pipeline runs and quality metrics are captured, tracked and reported

```sql
/* Stage 1: Bronze Table drop invalid rows */
CREATE STREAMING LIVE TABLE fire_account_bronze AS
  ( CONSTRAINT valid_account_open_dt EXPECT (account_dt is not null and (account_close_dt > account_open_dt)) ON VIOLATION DROP ROW
    COMMENT "Bronze table with valid account ids"
  )
SELECT * FROM fire_account_raw ...
```
Data pipeline observability

- High-quality, high-fidelity lineage diagram that provides visibility into how data flows for impact analysis
- Granular logging for operational, governance, quality and status of the data pipeline at a row level
- Continuously monitor data pipeline jobs to ensure continued operation
- Notifications using Databricks SQL
Declarative Pipelines CICD

-> “script” is conceptually executed from scratch

Ingestion (EL)  Downstream (T)

CREATE OR REFRESH MATERIALIZED VIEW
clean_data
AS SELECT ...
FROM raw_data

CREATE OR REFRESH MATERIALIZED VIEW
enriched_data
AS SELECT ...
FROM clean_data

Streaming Tables++  Materialized Views++

Data Lake

Kafka, Kinesis

CSV, JSON, TXT...

Spark

Streaming Analytics

BI & Reporting

Data Science & ML
Automated ETL development lifecycle

- Develop in environment(s) separate from production with the ability to easily test it before deploying – entirely in SQL
- Deploy and manage environments using parameterization
- Unit testing and documentation
- Enables metadata-driven ability to programatically scale to 100s of tables/pipelines dynamically
Automated ETL operations

- Reduce down time with automatic error handling and easy replay
- Eliminate maintenance with automatic optimizations of all Delta Live Tables
- Auto-scaling adds more resources automatically when needed.
Enhanced Autoscaling
Save infrastructure costs while maintaining end-to-end latency SLAs for streaming workloads

Problem
Optimize infrastructure spend when making scaling decisions for streaming workloads

- Built to handle streaming workloads which are spiky and unpredictable
- Shuts down nodes when utilization is low while guaranteeing task execution
- Only scales up to needed # of nodes

<table>
<thead>
<tr>
<th></th>
<th>AWS</th>
<th>Azure</th>
<th>GCP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Generally Available</td>
<td>Generally Available</td>
<td>Public Preview GA Coming Soon</td>
</tr>
</tbody>
</table>

Streaming source
Backlog monitoring
Utilization monitoring
No/Small backlog & low utilization
Scale down
Spark executors
Thank you