Project Title: ETL Pipeline for Home Insurance Claims & Policy Analytics

Business Goal:

A home insurance company wants to analyze **claims patterns, policy renewals, fraud detection, and regional risk**. This ETL project builds a pipeline that pulls data from multiple sources, processes it, creates features for downstream reporting and analytics (including ML), and manages it all through Git for production.

Tech Stack Overview:

Tool	Purpose
Azure Data Factory	Orchestration of data movement and transformation
Azure Data Lake Gen2	Central data storage in raw → cleaned → curated zones
Azure Databricks	Data transformation, feature engineering using Spark SQL
Azure DevOps / GitHub	Git-based deployment & version control
Power BI	Reporting and dashboards (optional)

Folder Structure in Azure Data Lake Storage Gen2

```
/home-insurance-data/
│
├── raw/
│   ├── policy_data.csv
│   ├── claims_data.json
│   └── property_info.csv
│
├── cleaned/
│   ├── policies_cleaned.parquet
│   ├── claims_cleaned.parquet
│
└── curated/
    ├── risk_features.parquet
    ├── fraud_signals.parquet
    └── renewal_predictions.parquet
```

Step-by-Step ETL Workflow

Step 1: Extract – Load Raw Data into Data Lake

Goal: Move source data (CSV, JSON, SQL tables) into the `raw` layer of the Data Lake.

Source Examples:

- Policy Management System (SQL Server)
- Claim Submission Portal (JSON exports)
- External property database (CSV, API)

ADF Pipeline: Extract Layer

- **Copy Activity 1:** SQL Server → Lake Gen2 `/raw/policy_data.csv`
- **Copy Activity 2:** REST API → Lake Gen2 `/raw/claims_data.json`
- **Copy Activity 3:** Blob or SFTP → `/raw/property_info.csv`

Pipeline Details:

- Use parameterized file paths
- Add date or file versioning in path (`raw/YYYY/MM/DD/`)

Output: All raw files stored in a structured format inside the lake.

Step 2: Clean & Normalize – Azure Databricks (PySpark)

Now we clean the data using Databricks. Here's an example with policy and claims data.

Policy Cleaning (Databricks Notebook)
python
```
df_policy = spark.read.option("header",
True).csv("abfss://data@insurance.dfs.core.windows.net/home-insurance-data/raw/policy_da
ta.csv")

df_policy_cleaned = df_policy \
    .dropna(subset=["policy_id", "customer_id", "premium_amount"]) \
    .withColumn("policy_start_date", to_date(col("start_date"), "yyyy-MM-dd")) \
    .withColumn("premium_amount", col("premium_amount").cast("double"))

df_policy_cleaned.write.mode("overwrite").parquet("abfss://data@insurance.dfs.core.windo
ws.net/home-insurance-data/cleaned/policies_cleaned.parquet")
```

Claims Cleaning
python
```
df_claims =
spark.read.json("abfss://data@insurance.dfs.core.windows.net/home-insurance-data/raw/cla
ims_data.json")

df_claims_cleaned = df_claims \
    .filter("status IS NOT NULL") \
    .withColumn("claim_amount", col("claim_amount").cast("double"))

df_claims_cleaned.write.mode("overwrite").parquet("abfss://data@insurance.dfs.core.windo
ws.net/home-insurance-data/cleaned/claims_cleaned.parquet")
```

Output: Normalized, typed, cleaned `.parquet` files in the `cleaned` zone.

Step 3: Transform & Feature Engineering – Spark SQL

Now build useful features for analytics and ML models.

Example Features:

- **Claim-to-premium ratio**
- **Average claims per zip code**
- **Number of policies per household**
- **Late renewal flag**
- **Claim filed within 30 days of new policy (fraud signal)**

Sample Spark SQL Transform:
sql
```
-- Join cleaned claims & policy data
SELECT
  p.policy_id,
  p.customer_id,
  p.premium_amount,
  c.claim_amount,
  DATEDIFF(c.claim_date, p.policy_start_date) AS days_since_start,
  CASE WHEN DATEDIFF(c.claim_date, p.policy_start_date) < 30 THEN 1 ELSE 0 END AS
potential_fraud,
  c.zip_code
FROM policies_cleaned p
JOIN claims_cleaned c ON p.policy_id = c.policy_id
```

Save this to curated zone:

python
```
features_df.write.mode("overwrite").parquet("abfss://data@insurance.dfs.core.windows.net
/home-insurance-data/curated/risk_features.parquet")
```

Output: Feature-rich datasets ready for BI and ML.

Step 4: ADF Pipeline with Databricks Activity

Build a final **ADF master pipeline**:

1. Copy raw data → raw zone
2. Run Databricks notebook → clean & transform
3. Output to curated zone

ADF → Add **Databricks Notebook Activity**

- Use dynamic parameters (`date`, `file_path`)
- Chain activities using **dependency conditions**

Final ADF pipeline runs full ETL chain.

Future Work : Reporting with Power BI

Connect Power BI to curated features (e.g., `risk_features.parquet`) to build dashboards:

- Top ZIP codes with high claims
- Policies with highest loss ratios
- Fraud detection alerts (visualized)
- Renewal rate trends