CARGO ATTRITIONAL CLAIMS ANALYSIS PROJECT

PROJECT OBJECTIVE:

To estimate ultimate claims reserves for cargo insurance using actuarial

methods, ensuring regulatory compliance and financial stability. This project addresses a critical business need - ensuring our company has

adequate reserves to pay future claims. We analysed 24 years of cargo

insurance data and used industry-standard actuarial methods to project

ultimate claims. The result: we need $17.8 million in additional reserves,

which represents a healthy 3.55% buffer."

BUSINESS PROBLEM:

✗ Insurance companies must set aside reserves for future claim payments

✗ Reserves too low → Risk of bankruptcy

✗ Reserves too high → Wasted capital

✗ Regulatory requirement to prove adequate reserves (Solvency II, IFRS 17)

KEY DELIVERABLES:

✓ Ultimate claims projection using Chain Ladder method

✓ IBNR (Incurred But Not Reported) reserve calculation

✓ Automated reporting and dashboards

✓ Executive summary for Board of Directors

SCOPE:

• Line of Business: Cargo Insurance (Marine & Aviation)

• Time Period: 24 accident years (2002-2025)

• Development Period: 282 months (23.5 years)

• Total Claims Analyzed: $502.6 Million

AGILE METHODOLOGY & REQUIREMENTS GATHERING

TITLE: Project Delivery Using Agile Scrum Framework

WHY AGILE?

✓ Complex project with evolving requirements

✓ Need for continuous stakeholder feedback

✓ Iterative development of analysis models

✓ Flexibility to adapt to data quality issues

PROJECT TIMELINE:

Sprint 0 (Planning): 1 week

Sprint 1-4 (Development): 8 weeks (2 weeks each)

UAT & Deployment: 1 week

Total Duration: 10 weeks

MEETING 1: Kick-off with Chief Actuary (Week 0)

Attendees:

• Chief Actuary (Project Sponsor)

• Senior Actuary (Subject Matter Expert)

• Head of Reserving

• Project Manager

• Data Analyst (Me)

Agenda:

1. Business context and regulatory drivers

2. Success criteria definition

3. Data availability and quality

4. Timeline and resource allocation

Key Questions Asked:

✓ What is the primary business question?

→ "How much should we reserve for cargo claims?"

✓ What level of granularity is required?

→ Monthly development periods, by accident year

✓ Which actuarial methods do you prefer?

→ Chain Ladder (primary), Bornhuetter-Ferguson (validation)

✓ What confidence level for projections?

→ 95% confidence level

✓ Who will consume the output?

→ Actuaries, Reserving team

Outputs:

• Project Charter (signed)

• High-level requirements document

• Stakeholder RACI matrix

MEETING 2: Technical Discovery with IT/Data Teams (Week 0)

Attendees:

• Database Administrator

• ETL Developer

• Data Warehouse Manager

• Data Governance Officer

• Data Analyst (Me)

Agenda:

1. Data source identification

2. Access permissions and security

3. Data refresh frequency

4. Known data quality issues

5. Technical constraints

Key Questions Asked:

✓ Where is claims data stored?

→ Claims DB (SQL Server), Policy DB (sql), Reinsurance DB (sql)

✓ What is the data refresh cycle?

→ Nightly batch load to Data Warehouse

✓ Are there known data quality issues?

→ Yes: Late reported claims, currency inconsistencies, null values

✓ What access do I need?

→ Read-only access to DW views, Windows Authentication

✓ Data retention and archival policies?

→ 7 years active, 20 years archived

Outputs:

• Data access request form (submitted)

• ETL documentation (reviewed)

• Data dictionary (received)

• Data quality report (analyzed)

MEETING 3: Requirements Workshop with Finance (Week 0)

Attendees:

• Financial Controller

* Reserving Head

• Reporting Analyst

• Data Analyst (Me)

Agenda:

1. Financial reporting requirements

2. Output format specifications

3. Regulatory compliance needs

4. Cash flow projection requirements

Key Questions Asked:

✓ What format do you need results in?

→ Excel for detailed analysis, PowerPoint for Board presentation

✓ Specific regulatory requirements?

→ IFRS 17 compliance, Solvency II capital calculations

✓ Tolerance for reserve volatility?

→ Prefer stable reserves year-over-year (< 10% change)

✓ When do you need results?

→ End of Q4 for year-end financial statements

Outputs:

• Reporting template specifications

• IFRS 17 mapping document

• Delivery schedule agreed

TOOL USED: Azure DevOps (TFS)

BACKLOG STRUCTURE:

Epic 1: Data Pipeline & ETL

├── Feature 1.1: Data Extraction from Source Systems

│ ├── PBI 1.1.1: Create SQL views for claims data

│ ├── PBI 1.1.2: Create SQL views for policy data

│ ├── PBI 1.1.3: Create SQL views for reinsurance data

│ ├── PBI 1.1.4: Implement incremental load logic

│ └── PBI 1.1.5: Add error handling and logging

│

├── Feature 1.2: Data Transformation & Cleansing

│ ├── PBI 1.2.1: Standardize date formats

│ ├── PBI 1.2.2: Handle null values and zeros

│ ├── PBI 1.2.3: Currency conversion to USD

│ ├── PBI 1.2.4: Duplicate detection and removal

│ └── PBI 1.2.5: Data validation rules

│

└── Feature 1.3: Data Loading to Mart

├── PBI 1.3.1: Create claims mart schema

├── PBI 1.3.2: Implement SCD Type 2 for history

├── PBI 1.3.3: Create aggregation tables

└── PBI 1.3.4: Build triangle view in mart

Epic 2: Actuarial Analysis & Modeling

├── Feature 2.1: Triangle Construction

│ ├── PBI 2.1.1: Extract triangle data from mart

│ ├── PBI 2.1.2: Handle missing values appropriately

│ ├── PBI 2.1.3: Validate triangle structure

│ └── PBI 2.1.4: Create triangle visualizations

│

├── Feature 2.2: Chain Ladder Implementation

│ ├── PBI 2.2.1: Calculate age-to-age factors

│ ├── PBI 2.2.2: Implement outlier detection

│ ├── PBI 2.2.3: Calculate cumulative dev factors

│ ├── PBI 2.2.4: Estimate tail factor

│ └── PBI 2.2.5: Project ultimate claims

│

└── Feature 2.3: IBNR Calculation & Validation

├── PBI 2.3.1: Calculate IBNR by accident year

├── PBI 2.3.2: Perform reasonability checks

├── PBI 2.3.3: Compare to prior year reserves

└── PBI 2.3.4: Document assumptions

Epic 3: Reporting & Visualization

├── Feature 3.1: Automated Report Generation

│ ├── PBI 3.1.1: Create executive summary template

│ ├── PBI 3.1.2: Generate technical report with code

│ ├── PBI 3.1.3: Create PowerPoint presentation

│ └── PBI 3.1.4: Export results to Excel

│

└── Feature 3.2: Dashboards & Visualizations

├── PBI 3.2.1: Create development heatmap

├── PBI 3.2.2: Create waterfall chart (IBNR bridge)

├── PBI 3.2.3: Create projection dashboard

└── PBI 3.2.4: Create executive dashboard

Epic 4: Documentation & Compliance

└── Feature 4.1: Documentation

├── PBI 4.1.1: Technical methodology document

├── PBI 4.1.2: User guide for actuaries

├── PBI 4.1.3: Audit trail documentation

└── PBI 4.1.4: GDPR compliance documentation

EXAMPLE PBI DETAIL (from TFS):

PBI 2.2.1: Calculate Age-to-Age Factors

Description:

Implement algorithm to calculate age-to-age development factors

across all accident years using median to ensure robustness.

Acceptance Criteria:

✓ Factors calculated for all consecutive development periods

✓ Median used instead of mean (outlier protection)

✓ Outliers (< 0.5 or > 5.0) excluded from calculation

✓ Missing values handled appropriately (skipped)

✓ Unit tests passing (>= 90% code coverage)

✓ Code review completed

✓ Documentation updated

Story Points: 5

Priority: High

Assigned To: Shweta Sandilya

Sprint: Sprint 2

Tags: Actuarial, Core-Logic, Chain-Ladder

Tasks:

* Write calculate\_ata\_factors() function
* Implement median calculation logic
* Add outlier detection (0.5-5.0 range)
* Handle NaN values
* Write unit tests
* Code review with Senior Actuary
* Update technical documentation

1. SPRINT PLANNING MEETING (Every 2 weeks, Monday 9 AM, 2 hours)

Attendees: Scrum Team + Product Owner (Chief Actuary)

Agenda:

• Review sprint goal

• Select PBIs from product backlog

• Break down PBIs into tasks

• Team commits to deliverables

• Capacity planning (account for holidays, training)

Example Sprint 2 Goal:

"Implement core Chain Ladder algorithm and calculate ultimate

projections for all accident years with >95% confidence."

Sprint 2 Commitment:

• 8 PBIs selected (34 story points)

• Expected velocity: 35 points (based on Sprint 1: 33 points)

• Team capacity: 80 hours (1 developer, 2 weeks, 100% allocation)

2. BACKLOG REFINEMENT MEETING (Weekly, Wednesday 2 PM, 1 hour)

Attendees: Scrum Team + Product Owner

Purpose: Groom upcoming PBIs for future sprints

Activities:

• Review new PBIs (from stakeholder feedback)

• Split large PBIs (>8 points) into smaller ones

• Add acceptance criteria to unrefined PBIs

• Re-estimate PBIs based on new information

• Re-prioritize backlog

Example Refinement Discussion:

Original PBI: "Handle data quality issues" (13 points - too large!)

Refined into 5 smaller PBIs:

• Handle null values in claim amounts (3 points)

• Standardize date formats across systems (2 points)

• Implement currency conversion (3 points)

• Detect and remove duplicates (2 points)

• Validate referential integrity (3 points)

3. DAILY STANDUP (Every day, 9:15 AM, 15 minutes)

Format: Stand-up meeting (literally standing!)

Three Questions Each Person Answers:

1. What did I complete yesterday?

2. What will I work on today?

3. Any blockers or impediments?

4. SPRINT REVIEW MEETING (End of sprint, Friday 2 PM, 1 hour)

Attendees: Scrum Team + Stakeholders (Actuary, Finance, IT)

Purpose: Demo completed work and get feedback

Agenda:

• Review sprint goal (achieved?)

• Demo completed PBIs (live demonstrations)

• Gather stakeholder feedback

• Accept/reject completed work

• Update product backlog based on feedback

We used Agile Scrum to manage this project, which allowed us to adapt quickly to data quality issues and changing stakeholder needs. We held extensive requirements gathering sessions with Actuarial, Finance, and IT teams to ensure we understood all perspectives. We tracked all work in Azure DevOps with clear Epic → Feature → PBI hierarchy, which gave stakeholders full visibility. Daily standups kept the team synchronized, and sprint reviews ensured continuous stakeholder feedback. This methodology was crucial for a complex analytical project like this

**Data Protection & Regulatory Compliance**

WHY THIS MATTERS:

• Handling sensitive insurance data (claims, policies, personal info)

• GDPR mandates strict data protection

• Corporate policy requires data security by design

1. LAWFULNESS, FAIRNESS & TRANSPARENCY

Processing necessary for actuarial analysis and regulatory

Implementation:

* Privacy notice provided to claimants
* Transparent about data usage
* Data Processing Impact Assessment (DPIA) completed

2. PURPOSE LIMITATION

Purpose: Actuarial analysis for reserve estimation ONLY

Implementation:

* Data NOT used for marketing or other purposes
* Access restricted to authorized personnel
* Clear documentation of processing purpose

3. DATA MINIMIZATION

Approach: Use ONLY what's necessary

Implementation:

* No personal identifiers (names, addresses) used
* Used aggregated data (accident year + development month)
* Removed fields not needed for analysis
* Anonymized ClaimID

4. ACCURACY

Measures:

* Data validation rules in ETL
* Reconciliation with source systems
* Quarterly data quality audits

5. STORAGE LIMITATION

Retention Policy:

• Active data: 7 years (regulatory requirement)

• Archived data: 20 years (litigation potential)

• Automated deletion after retention period

Implementation:

✓ TTL (Time To Live) set on data warehouse tables

✓ Automated archival process (nightly job)

✓ Deletion logs maintained for audit

6. INTEGRITY & CONFIDENTIALITY (Security)

See detailed security measures below

7. ACCOUNTABILITY

Documentation:

* Data Processing Record maintained
* Privacy Impact Assessment (PIA) completed
* Regular audits and reviews
* Breach notification process defined

LAYER 1: NETWORK SECURITY

* Data warehouse on premises (no internet access)
* VPN required for remote access
* Firewall rules (allow only specific IPs and ports)

LAYER 2: ACCESS CONTROL

* Windows Authentication with MFA (multi-factor)
* Role-Based Access Control (RBAC)
  + My role: "Analyst\_ReadOnly" (can read, cannot modify)
* Principle of Least Privilege

LAYER 5: DATA MASKING & ANONYMIZATION

✓ Dynamic Data Masking for non-privileged users

✓ Hash sensitive identifiers (ClaimID, Insured Name , Address, claimant name, address→ SHA-256)

LAYER 6: BACKUP & RECOVERY

* Nightly backups (encrypted)
* Off-site backup storage
* Tested recovery procedures (monthly)

1. NO HARDCODED CREDENTIALS

❌ BAD:

conn = pyodbc.connect('SERVER=sql01;UID=user;PWD=password123')

✅ GOOD:

Data base sensitive information like User Name and Password are stored in Hashicorp password vaults

2. PARAMETERIZED QUERIES (SQL Injection Prevention)

❌ BAD:

query = f"SELECT \* FROM Claims WHERE Year = {year}"

✅ GOOD:

query = "SELECT \* FROM Claims WHERE Year = ?"

cursor.execute(query, (year,))

3. ERROR HANDLING WITHOUT DATA LEAKAGE

❌ BAD:

except Exception as e:

print(f"Error: {e}") # May expose sensitive data

✅ GOOD:

except Exception as e:

logging.error(f"Error processing ClaimID {claim\_id}", exc\_info=True)

print("An error occurred. Contact support.")

4. SECURE TEMP FILES

✅ GOOD:

with tempfile.NamedTemporaryFile(delete=True) as tmp:

tmp.write(data)

# Process file

# Auto-deleted when closed

Data security and GDPR compliance were top priorities. I implemented multiple layers of protection: data minimization (used only aggregated, anonymized data), access control (read-only with MFA), and comprehensive audit logging. All processing aligns with GDPR's seven principles, and we completed a Data Protection Impact Assessment. The result: compliant analysis that protects customer privacy while meeting regulatory requirements.

DATA LIFECYCLE & ARCHITECTURE

End-to-End Data Architecture & ETL Pipeline

OVERVIEW: 7-Stage Data Lifecycle

Stage 1: Data Generation (Source Systems)

↓

Stage 2: Data Extraction (Nightly Jobs)

↓

Stage 3: Data Landing (PSA - Persistent Staging Area)

↓

Stage 4: Data Transformation (ETL Process)

↓

Stage 5: Data Loading (Data Warehouse Core)

↓

Stage 6: Data Mart Creation (Subject-Specific)

↓

Stage 7: Data Consumption (Analysis & Reporting)

STAGE 1: DATA GENERATION - SOURCE SYSTEMS

SOURCE SYSTEM 1: Claims Management System

Platform: Microsoft SQL Server 2019

Database: ClaimsDB

Tables:

• Claims (fact table)

• ClaimTransactions (payments, open, closed, reopened)

• ClaimStatus

Data Type: Structured (relational)

Refresh: Daily

SOURCE SYSTEM 2: Policy Administration System

Platform: Microsoft SQL Server 2019

Database: Subscribe DB

Tables:

• Policies (master data)

• PolicyCoverage (lines of business)

• PolicyPremiums (pricing)

Data Type: Structured (relational)

Refresh: Daily

SOURCE SYSTEM 3: Reinsurance System

Platform: RI Console

Database: RI Console DB

Tables:

• ReinsuranceContracts

• ReinsuranceClaims

• CedingTransactions

Data Type: Structured (relational)

Refresh: Weekly

DATA TYPE BREAKDOWN:

Structured Data: 95%

• Relational tables with defined schema

• Examples: ClaimID, AccidentDate, ClaimAmount, PolicyID

• Advantages: Easy to query, analyze, join

• Tools: SQL queries, pandas DataFrames

Semi-Structured Data: 5%

• Has some structure but not rigid schema

• Examples: JSON logs, XML files, CSV exports

• Challenges: Varying formats, nested structures

• Tools: JSON parsers, XML parsers

Unstructured Data: 0%

• No predefined structure

• Examples: Claim adjuster notes, PDF documents, images

• Challenges: Hard to analyze, requires NLP/OCR

• Tools: Text mining, computer vision (not used in this project)

STAGE 2: DATA EXTRACTION - NIGHTLY SCHEDULER PROCESS

SCHEDULER: SQL Server Agent Jobs (Automated)

Job Schedule:

• Frequency: Nightly

• Start Time: 11:00 PM (after business hours)

• End Time: 2:30 AM (before business starts)

• Retry: 3 attempts with 10-minute intervals

EXTRACTION METHODS:

Method 1: Full Extraction (Sunday only)

• Extract ALL data from source

• Used for: Reference data, small tables

• SQL Example:

SELECT \* FROM Claims WHERE LineOfBusiness = 'Cargo'

Method 2: Incremental Extraction (Monday-Saturday)

• Extract only NEW or CHANGED records

• Based on: LastModifiedDate column

• SQL Example:

SELECT \* FROM Claims

WHERE LastModifiedDate >= DATEADD(day, -1, GETDATE())

AND LineOfBusiness = 'Cargo'

Method 3: Change Data Capture (CDC) - Advanced

• Capture inserts, updates, deletes automatically

• Used for: High-volume transactional tables

• Lower impact on source system

EXTRACTION JOBS:

Job 1: Extract Claims (11:00 PM)

Source: ClaimsDB (SQL Server)

Destination: PSA.Claims\_RAW

Method: Incremental (LastModifiedDate)

Job 2: Extract Policies

Source: PolicyDB (SQL Server)

Destination: PSA.Policies\_RAW

Method: Full (reference data, small table)

Job 3: Extract Reinsurance

Source: ReinsuranceDB SQL Server)

Destination: PSA.Reinsurance\_RAW

Method: Incremental

DATA QUALITY CHECKS (Post-Extraction):

* Row count validation (compare source vs destination)
* NULL check on primary keys
* Date range validation (no future dates)
* Duplicate detection
* Email alert if validation fails

**STAGE 3: PSA (PERSISTENT STAGING AREA)**

PURPOSE:

Landing zone for raw data from source systems before transformation

CHARACTERISTICS:

• Exact copy of source data (no transformations)

• Append-only (historical snapshots preserved)

• Audit trail (LoadDate, LoadID columns)

SCHEMA EXAMPLE:

PSA.Claims\_RAW

Columns from source:

ClaimID, AccidentDate, ReportDate, ClaimAmount, PolicyID,

LineOfBusiness, ClaimStatus, ...

Audit columns (added):

PSA\_LoadDate DATETIME (when loaded)

PSA\_LoadID INT (batch ID)

PSA\_SourceSystem VARCHAR(50) (which source DB)

BENEFITS:

✓ Isolation from source systems (no re-querying needed)

✓ Point-in-time recovery possible

✓ Debugging (can trace data lineage)

✓ Reduces load on production systems

STAGE 4: ETL - DATA TRANSFORMATION

ETL JOB: Transform\_Claims

**TRANSFORMATION 1: DATA CLEANSING**

Issue 1: NULL Values

Problem: ClaimAmount has NULLs (data entry error)

Solution:

• Critical fields (ClaimID, AccidentDate): REJECT row, log error

• Optional fields (ClaimantEmail): Fill with 'UNCODED' or NULL

SQL:

DELETE FROM PSA.Claims\_RAW

WHERE ClaimID IS NULL OR AccidentDate IS NULL

Issue 2: Duplicates

Problem: Same ClaimID appears multiple times

Solution: Keep latest record (by LastModifiedDate)

SQL:

WITH RankedClaims AS (

SELECT \*,

ROW\_NUMBER() OVER (

PARTITION BY ClaimID

ORDER BY LastModifiedDate DESC

) AS rn

FROM PSA.Claims\_RAW

)

SELECT \* FROM RankedClaims WHERE rn = 1

Issue 3: Data Type Issues

Problem: ClaimAmount stored as VARCHAR (with commas, currency symbols)

Example: "$1,234,567.89"

Solution:

• Remove "$" and ","

• Convert to DECIMAL(18, 2)

SQL:

CAST(

REPLACE(REPLACE(ClaimAmount, '$', ''), ',', '')

AS DECIMAL(18, 2)

)

**TRANSFORMATION 2: DATA STANDARDIZATION**

Issue 1: Date Format Inconsistency

SQL Server: '2024-10-29' (YYYY-MM-DD)

Oracle: '29-OCT-24' (DD-MON-YY)

DB2: '10/29/2024' (MM/DD/YYYY)

Solution: Convert all to ISO 8601 (YYYY-MM-DD)

SQL:

CONVERT(DATE, AccidentDate, 120) -- SQL Server

TO\_DATE(AccidentDate, 'YYYY-MM-DD') -- Oracle

Issue 2: Currency Conversion

Problem: Claims in different currencies (USD, EUR, GBP, JPY)

Solution: Convert all to USD using ROE(Rate Of Exchange)

SQL:

CASE

WHEN Currency = 'USD' THEN ClaimAmount

WHEN Currency = 'EUR' THEN ClaimAmount \* 1.10

WHEN Currency = 'GBP' THEN ClaimAmount \* 1.27

WHEN Currency = 'JPY' THEN ClaimAmount \* 0.0067

ELSE NULL

END AS ClaimAmountUSD

Issue 3: Line of Business Naming

Problem: Inconsistent naming across systems

System A: "Cargo", "CARGO", "cargo"

System B: "Marine Cargo", "Vessel"

Solution: Mapping table + standardization

SQL:

UPDATE Claims

SET LineOfBusiness = 'Cargo'

WHERE LineOfBusiness IN ('CARGO', 'cargo', 'Marine Cargo', 'Ocean Cargo')

TRANSFORMATION 3: DATA ENRICHMENT

Derived Field 1: AccidentYear

SELECT YEAR(AccidentDate) AS AccidentYear

Derived Field 2: DevelopmentMonths

SELECT

DATEDIFF(MONTH, AccidentDate, EvaluationDate) AS DevMonths

Derived Field 3: IncurredAmount (Paid + Outstanding)

SELECT PaidAmount + OutstandingAmount AS IncurredAmount

Derived Field 4: Claim Age (in days)

SELECT DATEDIFF(DAY, AccidentDate, GETDATE()) AS ClaimAgeDays

**TRANSFORMATION 4: DATA INTEGRATION (Joins)**

Join Claims + Policies:

SELECT

c.ClaimID,

c.AccidentDate,

c.ClaimAmount,

p.PolicyNumber,

p.InsuredName,

p.Coverage,

p.Deductible

FROM PSA.Claims\_RAW c

LEFT JOIN PSA.Policies\_RAW p ON c.PolicyID = p.PolicyID

Join Claims + Reinsurance:

SELECT

c.ClaimID,

c.ClaimAmountUSD AS GrossAmount,

r.RecoveryAmount AS ReinsuranceRecovery,

c.ClaimAmountUSD - ISNULL(r.RecoveryAmount, 0) AS NetAmount

FROM Claims c

LEFT JOIN Reinsurance r ON c.ClaimID = r.ClaimID

TRANSFORMATION 5: DATA AGGREGATION

Aggregate to Triangle Format:

SELECT

YEAR(AccidentDate) AS AccidentYear,

DATEDIFF(MONTH, AccidentDate, EvaluationDate) AS DevMonths,

SUM(IncurredAmount) AS TotalIncurred,

COUNT(DISTINCT ClaimID) AS ClaimCount

FROM Claims

WHERE LineOfBusiness = 'Cargo'

GROUP BY

YEAR(AccidentDate),

DATEDIFF(MONTH, AccidentDate, EvaluationDate)

**TRANSFORMATION 6: DATA VALIDATION**

Validation Rule 1: No negative amounts

IF EXISTS (SELECT 1 FROM Claims WHERE ClaimAmount < 0)

BEGIN

RAISERROR('Negative claim amounts detected!', 16, 1)

ROLLBACK

END

Validation Rule 2: Accident date not in future

IF EXISTS (SELECT 1 FROM Claims WHERE AccidentDate > GETDATE())

BEGIN

RAISERROR('Future accident dates detected!', 16, 1)

ROLLBACK

END

Validation Rule 3: Referential integrity

IF EXISTS (

SELECT 1 FROM Claims c

WHERE NOT EXISTS (SELECT 1 FROM Policies p WHERE p.PolicyID = c.PolicyID)

)

BEGIN

-- Log orphan claims for review

INSERT INTO ErrorLog (Message, Severity)

VALUES ('Orphan claims without policies found', 'Warning')

END

ERROR HANDLING:

* Try-Catch blocks in SQL stored procedures
* Errors logged to ETL\_ErrorLog table
* Email alerts on critical errors
* Rollback on validation failures
* Retry logic for transient errors

STAGE 5: DATA WAREHOUSE CORE (Star Schema)

ARCHITECTURE:

FACT TABLE: Fact\_Claims Add image for this

ClaimKey BIGINT (Surrogate Key)

ClaimID VARCHAR(50) (Business Key)

AccidentDateKey INT (FK to Date Dimension)

ReportDateKey INT (FK to Date Dimension)

EvalDateKey INT (FK to Date Dimension)

PolicyKey BIGINT (FK to Policy Dimension)

LineOfBusinessKey INT (FK to LOB Dimension)

IncurredAmount DECIMAL(18, 2)

PaidAmount DECIMAL(18, 2)

OutstandingAmount DECIMAL(18, 2)

ClaimCount INT (always 1 for additive)

DevMonths INT (calculated field)

LoadDate DATETIME

SourceSystem VARCHAR(50)

DIMENSION: Dim\_Date

DateKey INT (PK, YYYYMMDD format: 20241029)

FullDate DATE

Year INT

Quarter INT

Month INT

MonthName VARCHAR(20)

DayOfWeek INT

IsWeekend BIT

IsHoliday BIT

FiscalYear INT

FiscalQuarter INT

DIMENSION: Dim\_Policy

PolicyKey BIGINT (PK, Surrogate)

PolicyID VARCHAR(50) (Business Key)

PolicyNumber VARCHAR(50)

InsuredName VARCHAR(200) (Anonymized)

Coverage VARCHAR(100)

Deductible DECIMAL(18, 2)

SCD\_StartDate DATE (for history tracking)

SCD\_EndDate DATE

SCD\_IsCurrent BIT (1 = current, 0 = historical)

DIMENSION: Dim\_LineOfBusiness

LOBKey INT (PK)

LOBCode VARCHAR(10)

LOBName VARCHAR(100)

LOBCategory VARCHAR(50)

SCD TYPE 2 (Slowly Changing Dimensions):

Used for Dim\_Policy to track historical changes

Example:

Policy ABC123 changed coverage on 2023-06-01

Before change:

PolicyKey | PolicyID | Coverage | SCD\_StartDate | SCD\_EndDate | IsCurrent

1001 | ABC123 | Basic | 2020-01-01 | 2023-05-31 | 0

1002 | ABC123 | Premium | 2023-06-01 | 9999-12-31 | 1

Benefit: Can analyze claims using policy attributes as of accident date

**STAGE 6: DATA MART CREATION**

DATA MART: Actuarial\_Mart (Subject: Actuarial Analysis)

Purpose: Optimized for actuarial queries (triangle analysis, IBNR)

MART TABLE: Mart\_Claims\_Triangle

CREATE VIEW vw\_Actuarial\_Claims\_Triangle AS

SELECT

-- Dimensions

dd.Year AS AccidentYear,

f.DevMonths,

-- Measures (Aggregated)

SUM(f.IncurredAmount) AS TotalIncurred,

SUM(f.PaidAmount) AS TotalPaid,

SUM(f.OutstandingAmount) AS TotalOutstanding,

COUNT(DISTINCT f.ClaimID) AS ClaimCount,

-- Metadata

dlob.LOBName AS LineOfBusiness,

MAX(f.LoadDate) AS LastRefreshDate

FROM Fact\_Claims f

INNER JOIN Dim\_Date dd ON f.AccidentDateKey = dd.DateKey

INNER JOIN Dim\_LineOfBusiness dlob ON f.LineOfBusinessKey = dlob.LOBKey

WHERE

dlob.LOBName = 'Cargo'

AND f.DevMonths >= 0

AND f.IncurredAmount > 0

AND dd.Year >= 2002

GROUP BY

dd.Year,

f.DevMonths,

dlob.LOBName

BENEFITS OF DATA MART:

✓ Pre-aggregated data (faster queries)

✓ Subject-specific (only relevant data)

✓ Simplified schema (easier for analysts)

✓ Can grant access without exposing full DW

INDEXING STRATEGY:

CREATE INDEX IX\_Triangle\_Year\_Dev

ON Mart\_Claims\_Triangle (AccidentYear, DevMonths)

MATERIALIZATION:

• View refreshed nightly (after ETL)

• Can materialize as table for very large datasets

• Partition by AccidentYear for performance

**STAGE 7: DATA CONSUMPTION**

Python Analysis (My Project!)

* Load the extracted excel in python in Python using pandas
* Created claims triable from the raw data
* Applied Chain Ladder algorithm
* Generated automated reports and visualizations
* Advantage: Reproducible, auditable, version-controlled

We implemented a comprehensive data architecture following best practices. Data flows from three source systems through nightly extraction jobs into a Persistent Staging Area. The ETL process handles data quality issues like NULL values, duplicates, date format inconsistencies, and currency conversions. Cleansed data is loaded into a star schema Data Warehouse. An Actuarial Mart provides cleaned and optimised data for the analysis Finally, data is consumed through Power BI dashboards, Excel exports, and Python analysis. This architecture ensures data quality, security, and performance

**PROJECT IMPLEMENTATION & METHODOLOGY**

Actuarial Methodology - Chain Ladder Technique

WHY CHAIN LADDER METHOD?

DECISION CRITERIA:

* Industry Standard:
* Simple & Transparent:
* Auditable: Clear audit trail from data to results
* Stable: Minimal volatility year-over-year
* Data Requirements: Works with standard triangle data

CHOSEN: Chain Ladder (with median, not mean)

Rationale: Best balance of simplicity, robustness, and acceptability

code to create triangle from raw data

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen with white text

AI-generated content may be incorrect.

CHAIN LADDER METHODOLOGY - DETAILED

A black rectangle with white text

AI-generated content may be incorrect.

STEP 1: CONSTRUCT CLAIMS DEVELOPMENT TRIANGLE

Input Data:

• Accident Years: 2002 to 2025 (24 years)

• Development Months: 1 to 282 (23.5 years)

• Measure: Cumulative Incurred Amount (Paid + Outstanding)

Triangle Structure:

Properties:

• Upper-left: Fully developed (older years, later months)

• Lower-right: Missing data (recent years, early months)

• Triangle shape: Each row has fewer data points as years get recent

STEP 2: CALCULATE AGE-TO-AGE (ATA) FACTORS

For each pair of consecutive development months, calculate:

ATA Factor = Claims at Month (n+1) / Claims at Month n

A computer screen shot of a program

AI-generated content may be incorrect.

Repeat for all development periods:

1→2, 2→3, 3→4, ..., 281→282

STEP 3: SELECT DEVELOPMENT FACTORS

Options for selection:

a) Median (our choice - robust to outliers)

b) Mean (average - affected by outliers)

c) Weighted average (by volume)

d) Latest year only (risky - single data point)

e) Average of latest 3 years (more weight on recent patterns)

A screenshot of a computer program

AI-generated content may be incorrect.

We chose MEDIAN because:

✓ Robust to outliers (catastrophic claims won't skew results)

✓ Standard actuarial practice

✓ Performed well in back-testing

Outlier Exclusion:

• Exclude factors < 0.5 or > 5.0 (likely data errors)

• Log excluded values for review

STEP 4: ESTIMATE TAIL FACTOR

Problem: Triangle ends at 282 months, but claims may develop beyond

Solution: Estimate a "tail factor" for development to ultimate

Method:

• Review last 5 ATA factors

• If all close to 1.0 (< 5% growth): Use conservative tail = 1.01

• If still growing: Use geometric mean of last 3, dampened by 50%

A computer screen shot of a program code

AI-generated content may be incorrect.

Example:

STEP 5: CALCULATE CUMULATIVE DEVELOPMENT FACTORS (CDFs)

A computer screen with text

AI-generated content may be incorrect.

CDF = Product of all ATA factors from current period to ultimate

Example:

CDF at Month 60 = ATA(60→72) × ATA(72→84) × ... × ATA(281→282) × Tail

= 1.05 × 1.03 × 1.02 × ... × 1.001 × 1.01

≈ 1.15

Interpretation:

Claims at month 60 will grow by 15% to reach ultimate

Calculation (working backwards from tail):

Period ATA Factor CDF (cumulative product)

──────────────────────────────────────────────────

282 (tail) 1.0100

281 1.0010 1.0100 × 1.0010 = 1.0110

280 1.0012 1.0110 × 1.0012 = 1.0122

...

60 1.0500 (result of all multiplications) = 1.1500

...

1 1.0220 (result of all multiplications) = 25.3400

STEP 6: PROJECT ULTIMATE CLAIMS

For each accident year:

1. Find latest reported claims value

2. Identify development period of that value

3. Look up CDF for that period

4. Calculate: Ultimate = Latest × CDF

Example (Accident Year 2020):

Latest Period: Month 60

Latest Value: $12,456,789

CDF at Month 60: 1.0478

Ultimate = $12,456,789 × 1.0478 = $13,052,301

STEP 7: CALCULATE IBNR

IBNR = Ultimate - Latest

Example (Accident Year 2020):

IBNR = $13,052,301 - $12,456,789 = $595,512

Interpretation:

We need $595,512 in additional reserves for 2020 claims

STEP 8: AGGREGATE RESULTS

Sum across all accident years:

Total Latest: $502,600,000

Total Ultimate: $520,400,000

Total IBNR: $17,800,000

IBNR Ratio: $17.8M / $502.6M = 3.55%

CLASS STRUCTURE:

class CargoTriangleAnalyzer:

def \_\_init\_\_(self, input\_file):

self.input\_file = input\_file

self.triangle = None

self.ata\_factors = None

self.projections = None

def load\_and\_parse\_triangle(self):

# Load Excel, extract metadata and triangle data

# Returns: DataFrame with years as columns, dev months as rows

def calculate\_ata\_factors(self):

# For each consecutive period, calculate median factor

# Exclude outliers (< 0.5 or > 5.0)

# Returns: Dictionary {period: factor}

def calculate\_tail\_factor(self):

# Estimate tail based on last 5 factors

# Returns: Float (e.g., 1.01)

def project\_ultimate\_claims(self):

# Calculate CDFs, apply to each year

# Returns: DataFrame with projections

def create\_visualizations(self):

# Generate heatmaps, waterfalls, dashboards

def save\_results\_to\_excel(self):

# Export results to Excel workbook

def generate\_word\_report(self):

# Create automated Word document with analysis

KEY DESIGN PATTERNS:

1. Object-Oriented: Encapsulate related functionality

2. Single Responsibility: Each method has one clear purpose

3. DRY (Don't Repeat Yourself): Reusable functions

4. Defensive Programming: Validate inputs, handle errors

5. Logging: Track progress and issues

PYTHON LIBRARIES USED:

• pandas: Data manipulation (triangle, factors, projections)

• numpy: Numerical calculations (median, product, log)

• matplotlib: Visualizations (charts, plots)

• seaborn: Statistical visualizations (heatmaps)

• openpyxl: Excel file handling (read/write)

• python-docx: Word document generation

• datetime: Date handling and formatting

CODE QUALITY PRACTICES:

✓ Docstrings: Every function documented

✓ Type hints: Function signatures include types (Python 3.10+)

✓ Error handling: Try-except blocks with logging

✓ Unit tests: Coverage >85% (pytest framework)

✓ Code review: Reviewed by Senior Actuary

✓ Version control: Git with meaningful commit messages

✓ PEP 8 compliance: Consistent code style

VALIDATION 1: Reconciliation to Source Data

✓ Total claims in triangle match source data (within 0.01%)

✓ Row counts consistent with expectations

VALIDATION 2: Reasonability Tests

✓ ATA factors between 0.5 and 5.0 (exclude outliers)

✓ CDFs decrease as development period increases (mature → 1.0)

✓ IBNR positive for all years (claims grow, don't shrink)

✓ % Developed increases with age (old years >95%, recent <60%)

VALIDATION 3: Comparison to Prior Year

✓ Ultimate claims for mature years stable (<5% change)

✓ IBNR ratio consistent (3.5% this year vs 3.8% last year)

✓ Significant changes investigated and explained

VALIDATION 4: Sensitivity Analysis

✓ Vary tail factor (1.00 to 1.05): IBNR changes by ±$2M

✓ Exclude last year data: IBNR changes by ±$1M

✓ Use mean instead of median: IBNR changes by +$5M (outlier effect)

→ Conclusion: Results robust to methodology choices

VALIDATION 5: Actuarial Review

✓ Methodology reviewed by qualified actuary (FCAS credential)

✓ Results reviewed for reasonability

✓ Sign-off obtained before presentation to CFO

TALKING POINTS:

"I chose the Chain Ladder method because it's the industry standard, widely

accepted by regulators and auditors, and strikes the right balance between

simplicity and accuracy. The implementation uses median factors for

robustness to outliers, proper handling of missing data in the triangle,

and a conservative tail factor for development beyond our observed data.

I implemented the algorithm in Python using pandas for data manipulation and

created an object-oriented class structure for maintainability. The results

were validated through reconciliation, reasonability tests, sensitivity

analysis, and actuarial review before presentation."

FINDING 1: ADDITIONAL RESERVES REQUIRED

Total Reported Claims: $502.6 Million

Projected Ultimate Claims: $520.4 Million

IBNR Reserve Needed: $17.8 Million

IBNR as % of Reported: 3.55%

Interpretation:

✓ Reserves are in good shape (3.55% is healthy)

✓ Not materially under-reserved (would be concerning if >10%)

✓ Not materially over-reserved (efficient capital allocation)

Comparison to Prior Year:

Last Year IBNR Ratio: 3.8%

This Year IBNR Ratio: 3.55%

Change: -0.25 percentage points (IMPROVEMENT)

Reason: Claims developing faster than expected (positive news!)

FINDING 2: CLAIMS MATURITY

Average % Developed: 93.21%

By Accident Year Band:

2002-2010 (Old): 99.5% developed (fully mature)

2011-2020 (Middle): 95.2% developed (very mature)

2021-2023 (Recent): 75.8% developed (still developing)

2024-2025 (Latest): 45.3% developed (immature)

Interpretation:

✓ High maturity = Low uncertainty

✓ Projections for old years very reliable

✓ Projections for recent years have more uncertainty

FINDING 3: CONSISTENT DEVELOPMENT PATTERNS

Pattern Observed:

• Early months (1-12): High volatility, rapid growth

• Middle months (12-60): Stable, predictable growth (~5-10% annually)

• Late months (60+): Minimal growth (<1% annually)

Anomaly Detected:

• Spike in development at Month 5-6 (factor 1.82)

• Explanation: Reporting lag - cargo claims often reported 6 months late

• This is expected and consistent across all years

Validation:

✓ Patterns consistent year-over-year

✓ No unexpected changes in development

✓ Historical data reliable predictor of future behavior

FINDING 4: IBNR CONCENTRATION

Top 5 Years by IBNR:

2024: $7.9M (44% of total IBNR)

2023: $5.3M (30% of total IBNR)

2025: $5.2M (29% of total IBNR)

────────────────────────────────

Total (Top 3): $18.4M (103% of total - older years have negative IBNR)

Interpretation:

✓ IBNR concentrated in recent years (expected)

✓ Older years slightly over-reserved (releasing reserves)

✓ Net effect: $17.8M additional reserves needed

IMPLICATION 1: FINANCIAL IMPACT

• Need to increase reserves by $17.8M

• Impact on equity: -$17.8M

• Impact on capital ratio: Minimal (-0.2 percentage points)

• No solvency concerns (still well above regulatory minimum)

IMPLICATION 2: PRICING

• Current pricing appears adequate (IBNR ratio 3.55% is healthy)

• No need for rate increases at this time

• Monitor future development for trends

IMPLICATION 3: REGULATORY REPORTING

• Results support Year-End 2025 financial statements

• Adequate reserves for Solvency II compliance

• IFRS 17 discounting requirements separate (not covered in this analysis)

IMPLICATION 4: CASH FLOW

• $17.8M will be paid out over next 5-10 years

• Projected payment pattern:

Year 1: $6.2M (35%)

Year 2: $4.5M (25%)

Year 3: $3.2M (18%)

Year 4: $2.1M (12%)

Year 5+: $1.8M (10%)

• Treasury can plan investments accordingly

RECOMMENDATION 1: IMMEDIATE ACTIONS

1. Increase cargo reserves by $17.8M in Q4 2025 financials

2. Notify Board of Directors of reserve adjustment

3. Update external auditors on methodology and results

RECOMMENDATION 2: ONGOING MONITORING

1. Re-run analysis quarterly to monitor development

2. Investigate if actual development deviates >5% from projection

3. Update methodology annually based on latest data

RECOMMENDATION 3: METHODOLOGY ENHANCEMENTS

1. Implement Bornhuetter-Ferguson as alternative method (validation)

2. Add formal uncertainty quantification (Mack's formula, bootstrapping)

3. Consider stochastic modeling for capital adequacy (1-in-200 scenarios)

RECOMMENDATION 4: OPERATIONAL IMPROVEMENTS

1. Investigate reporting lag at Month 5-6 (can we reduce it?)

2. Improve data quality (reduce nulls, duplicates)

3. Enhance claims handling for faster settlement (reduce reserves)

RECOMMENDATION 5: AUTOMATION & SCALABILITY

1. Deploy this analysis to production (scheduled monthly run)

2. Extend to other lines of business (Property, Liability, etc.)

3. Build interactive Power BI dashboard for real-time monitoring

IMMEDIATE (Next 2 Weeks):

□ Present findings to CFO and Chief Actuary

□ Finalize reserve adjustment ($17.8M)

□ Update financial systems with new reserve balances

□ Notify external auditors

SHORT-TERM (Next Quarter):

□ Implement quarterly monitoring process

□ Document methodology for audit

□ Train junior actuaries on process

□ Archive code and data for future reference

LONG-TERM (Next Year):

□ Extend analysis to all lines of business

□ Implement alternative methods (BF, Cape Cod)

□ Build Power BI dashboard for executives

□ Integrate with financial planning systems

PROJECT: Cargo Attritional Claims Analysis - Reserve Estimation

METHODOLOGY: Chain Ladder technique with median factors

DATA: 24 years of cargo claims (2002-2025), 282 months development

KEY RESULTS:

• Total Reported Claims: $502.6 Million

• Projected Ultimate Claims: $520.4 Million

• IBNR Reserve Required: $17.8 Million (3.55%)

• Average Claims Maturity: 93.21% developed

CONCLUSION:

Reserves are adequate. Need $17.8M increase, which represents a healthy

3.55% buffer. Results validated through multiple methods and actuarial

review. No material concerns.

RECOMMENDATION:

Approve $17.8M reserve increase. Implement quarterly monitoring. Extend

methodology to other lines of business.

"Our analysis of 24 years of cargo claims data using the industry-standard

Chain Ladder method shows we need $17.8 million in additional reserves,

which is 3.55% of reported claims. This indicates our reserves are in good

shape - not too high, not too low. The analysis was validated through

multiple checks and reviewed by a qualified actuary. We recommend approving

this reserve increase for Q4 financials and implementing quarterly

monitoring going forward. This positions us well for regulatory compliance

and financial stability."

APPENDIX: Additional Materials Available Upon Request

• Code Repository with Unit Tests

• Detailed Triangle Data (Excel)

• Sensitivity Analysis Results

• Actuarial Review Sign-off

• GDPR Compliance Documentation

• Data Architecture Diagrams

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect..

A screen shot of a computer

AI-generated content may be incorrect.

A graph of a graph and a graph of a graph

AI-generated content may be incorrect.

A close-up of a graph

AI-generated content may be incorrect.

A collage of graphs

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

A graph of a number of blue bars

AI-generated content may be incorrect.

A graph of different colored lines

AI-generated content may be incorrect.

A group of graphs showing different types of graphs

AI-generated content may be incorrect.

A collage of graphs

AI-generated content may be incorrect.

A graph of different sizes and colors

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A computer code on a black background

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.