Visit Limit Algorithm - Organized Document

Presenting an Automated Solution for Comparative Analysis of Visit Limits

Benefits of the Algorithm

1. Efficiency: Automates the analysis, reducing the need for manual review.
2. Scalability: Handles large datasets typical of insurance companies.
3. Cost-Effective: Minimizes labor costs associated with manual reviews.
4. Accuracy: Ensures consistent and reliable results by eliminating human error.

Algorithm Overview

The algorithm consists of three main steps:

1. Database Preparation
2. Data Extraction
3. Parallel Processing

Step 1: Database Preparation

Indexing:

Purpose: Speed up data retrieval by creating indexes on key columns.

Relevant Columns:

- Claim Number: ClaimNumber
- Date of Service: DateOfService
- Procedure Code (CPT): CPT

- Year: Year

Partitioning:

Purpose: Divide the database into smaller, more manageable pieces based on criteria like year.

Relevant Columns:

- Year: Year

Step 2: Data Extraction

Optimized Queries:

Purpose: Efficiently extract relevant data segments.

Relevant Columns:

- Year: Year
- State: State
- Individual: Individual
- Date of Service: DateOfService
- Claim Number: ClaimNumber
- Billed Amount: BilledAmount
- Payment: Payment
- Deductible: Deductible
- Reason Code: ReasonCode
- Plan Category: PlanCategory
- Product: Product
- Plan Type: PlanType
- Mental Health/Substance Use Disorder ICD-10 Code: MH_SUD_ICD10

- Medical/Surgical ICD-10 Code: Med_Surg_ICD10
- Procedure Code (CPT): CPT
- Number of Visits in Year 1: VisitsYear1
- Number of Visits in Year 2: VisitsYear2
- Number of Visits in Year 3: VisitsYear3

Step 3: Parallel Processing with Apache Spark

Apache Spark:

Purpose: A powerful tool designed for processing large datasets in parallel.

Relevant Columns:

- Year: Year
- State: State
- Individual: Individual
- Date of Service: DateOfService
- Claim Number: ClaimNumber
- Billed Amount: BilledAmount
- Payment: Payment
- Deductible: Deductible
- Reason Code: ReasonCode
- Plan Category: PlanCategory
- Product: Product
- Plan Type: PlanType
- Mental Health/Substance Use Disorder ICD-10 Code: MH_SUD_ICD10
- Medical/Surgical ICD-10 Code: Med_Surg_ICD10

Visit Limit Algorithm - Organized Document

- Procedure Code (CPT): CPT
- Number of Visits in Year 1: VisitsYear1
- Number of Visits in Year 2: VisitsYear2
- Number of Visits in Year 3: VisitsYear3

Explanation of Flags with Column Descriptions

1. Flag A: Reprocessed Claims

Purpose: Identifies claims that have been processed multiple times.

Relevant Columns:

- Claim Number: ClaimNumber
- Date of Service: DateOfService

Example: If a patient has multiple claim lines for the same date of service, it may indicate reprocessing due to visit limits.

2. Flag B: Front-Loading Services

Purpose: Detects if services are predominantly used early in the year.

Relevant Columns:

- Procedure Code (CPT): CPT
- Date of Service: DateOfService

Example: If all therapy sessions for a patient occur between January and June, it suggests front-loading to use benefits before limits apply.

3. Flag C: Diagnosis Switching

Purpose: Identifies if diagnoses switch from mental health to medical conditions.

Visit Limit Algorithm - Organized Document

Relevant Columns:

- Individual: Individual

- Mental Health/Substance Use Disorder ICD-10 Code: MH_SUD_ICD10

- Medical/Surgical ICD-10 Code: Med_Surg_ICD10

Example: A patient initially diagnosed with an eating disorder switches to diabetes management, potentially to bypass visit limits on mental health services.

4. Flags D, E, F, G: Consistent Visit Counts

Purpose: Detects consistent numbers of approved visits over multiple years.

Relevant Columns:

- Individual: Individual

- Number of Visits in Year 1: VisitsYear1

- Number of Visits in Year 2: VisitsYear2

- Number of Visits in Year 3: VisitsYear3

Example: If a patient consistently receives 8 approved therapy sessions per year, it may indicate a visit limit.

Example Implementation

Database Preparation

Indexing Example:

Creating an index on ClaimNumber for faster retrieval:

```sql
CREATE INDEX idx_claim_number ON Claims(ClaimNumber);
```

Visit Limit Algorithm - Organized Document

```

Partitioning Example:

Partitioning the database by year:

```sql
CREATE TABLE ClaimsPartitioned PARTITION BY RANGE (Year) (
PARTITION p2021 VALUES LESS THAN (2022),
PARTITION p2022 VALUES LESS THAN (2023),
PARTITION p2023 VALUES LESS THAN (2024)
) AS SELECT * FROM Claims;
```

Data Extraction

Optimized Query Example:

Extracting data for the year 2021:

```python
import pandas as pd
import sqlite3

def extract_data(year):
    conn = sqlite3.connect('cigna_claims.db')
    query = f"""
    SELECT * FROM ClaimsPartitioned WHERE Year = {year};
    """
```
```

```
    data = pd.read_sql_query(query, conn)
    conn.close()
    return data

data_2021 = extract_data(2021)
```

Parallel Processing with Apache Spark

Initializing Spark:

```python
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Claims Analysis Optimization") \
    .config("spark.sql.shuffle.partitions", "8") \
    .getOrCreate()
```

Loading and Analyzing Data:

Loading data into Spark and performing analysis:

```python
claims_df = spark.read.csv("claims_data.csv", header=True, inferSchema=True)

# Flag A: Identify reprocessed claims
```

```
reprocessed_claims = claims_df.groupBy("ClaimNumber", "DateOfService").count().filter("count > 1")

# Save results
reprocessed_claims.write.csv("reprocessed_claims_output.csv")

spark.stop()
```