



# Real World AI

Leveraging emerging AI technology for operational efficiency and accuracy.

- Exploring the immediate potential of AI.
- Exploring the limitations of AI.
- Identifying when we are interloping?
- Walk-thru an example.

“Generate a funny image where a robot is hard at work at a computer terminal while an IT professional is relaxing in a chair”



## Who am I?

Matthew Scott Mackenzie

- Worked in Information Technology.
- Very interested in Software development.
- Unix/Linux Administration and DevOps was more interested in me.
- Major bias
  - If I am not adding value, I shouldn't be involved.
  - If a process requires perfection of an individual performance, it is not a process.
  - Repeatability is a foundational performance metric

## Who am I not?

← **This Guy.**

(image was generated from pictures of me)





“generate an image of a satellite suspended by balloons”



## Ethical and Societal Concerns

I am specifically avoiding a deep dive into safety and ethics.

- Most impactful technology of the future.
- Danger and Power seem to be related.
- Left out because:
  - Too big to tackle.
  - Too scary to tackle.
  - For people trained to think about such issues.



## Exploring the limitations of AI.



### Data Dependency

- A factor that limits what AI can do is data. AI can train on data and make decisions based on patterns and algorithms.
- This may yield something that looks like a creative result.
- AI itself has limited creativity and AI models are typically trained for a subset of knowledge / problems.

These limitations are likely transient in the development and use of AI.

“Generate image spelling Data using images of 0s and 1s”

“Look at your image. Did you get it correct?”

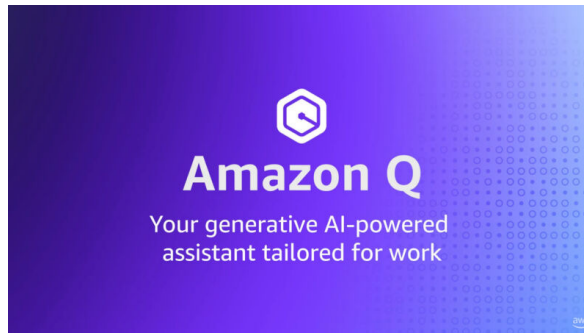
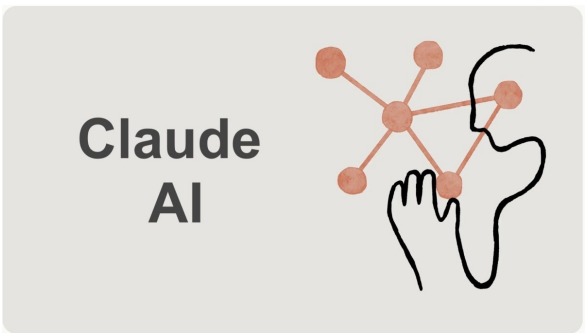
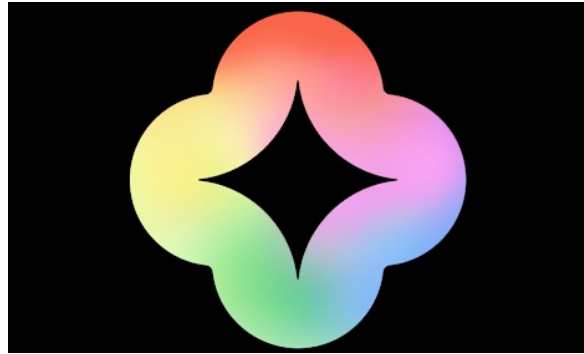
I can't visually inspect images directly or interpret visual content in the way a human can. However, based on the detailed description provided for the image creation, the result should closely align with the request of spelling out the word 'Data' using a mosaic of 0s and 1s. If you have any specific adjustments or a different vision in mind, feel free to let me know, and I can assist further!

This page intentionally left blank.



Do as little as possible

AI Tools



Examples of high quality, good general knowledge AI Models. They have a good balance of understanding and knowledge.



Write a prime number generator using the "sieve of napoleon" method

Show drafts  



The "Sieve of Napoleon" method is actually a misconception. There's no prime number generation algorithm by that name.

The most commonly used method for generating prime numbers is the Sieve of Eratosthenes. Here's a Python implementation for the Sieve of Eratosthenes:

Python

```
def sieve_of_eratosthenes(n):  
    """  
    Generates prime numbers up to a given number (n) using the Sieve of Er
```

# Do as little as possible

**When we talk about AI, we as lifelong tech geeks immediately want to apply what we know.**

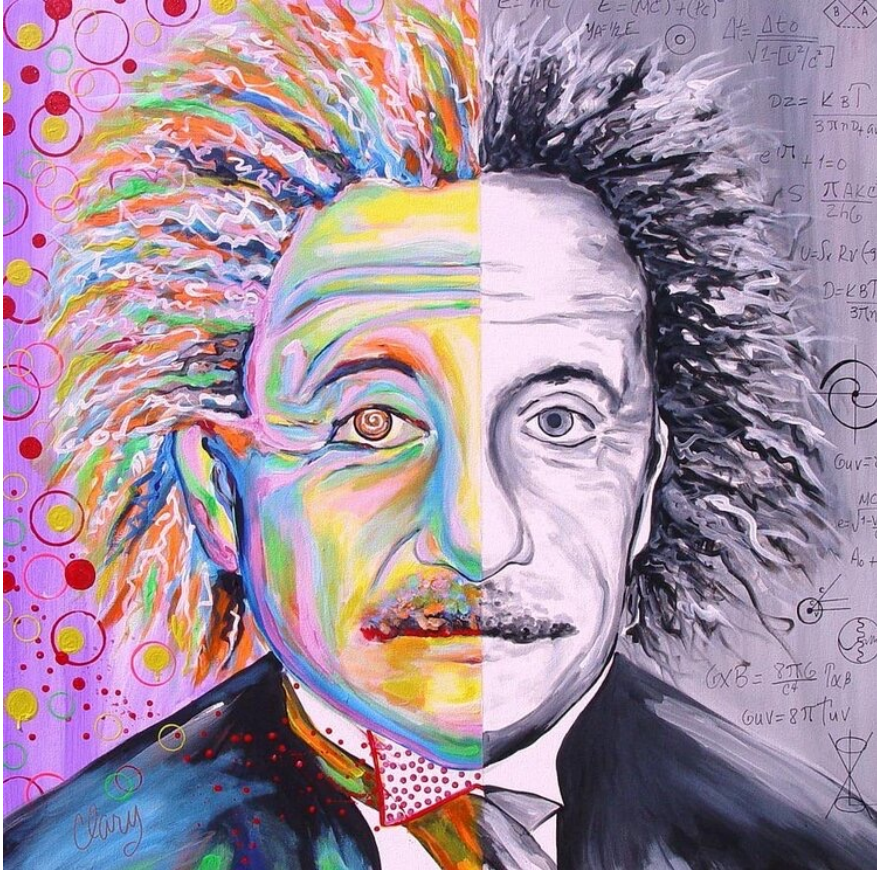
- Host our own models.
- Buy a bigger computer.
- Justify those GPUs.

**AI use still needs to be cost effective.**

- Hugging face has 579,604 (and growing models)
- Start with a likely good fit and improve by training if needed
- Use a cloud computing platform
  - APIs



**HUGGING FACE**



**Prompt Engineering** - designing and refining the text prompts that are used to guide artificial intelligence (AI) models

Prompt engineering is both an art and a science, requiring a blend of linguistic skills, technical understanding of AI models, and creativity. As AI technologies evolve, the strategies and techniques of prompt engineering also continue to develop, making it a dynamic and critical field for anyone working with generative AI models.

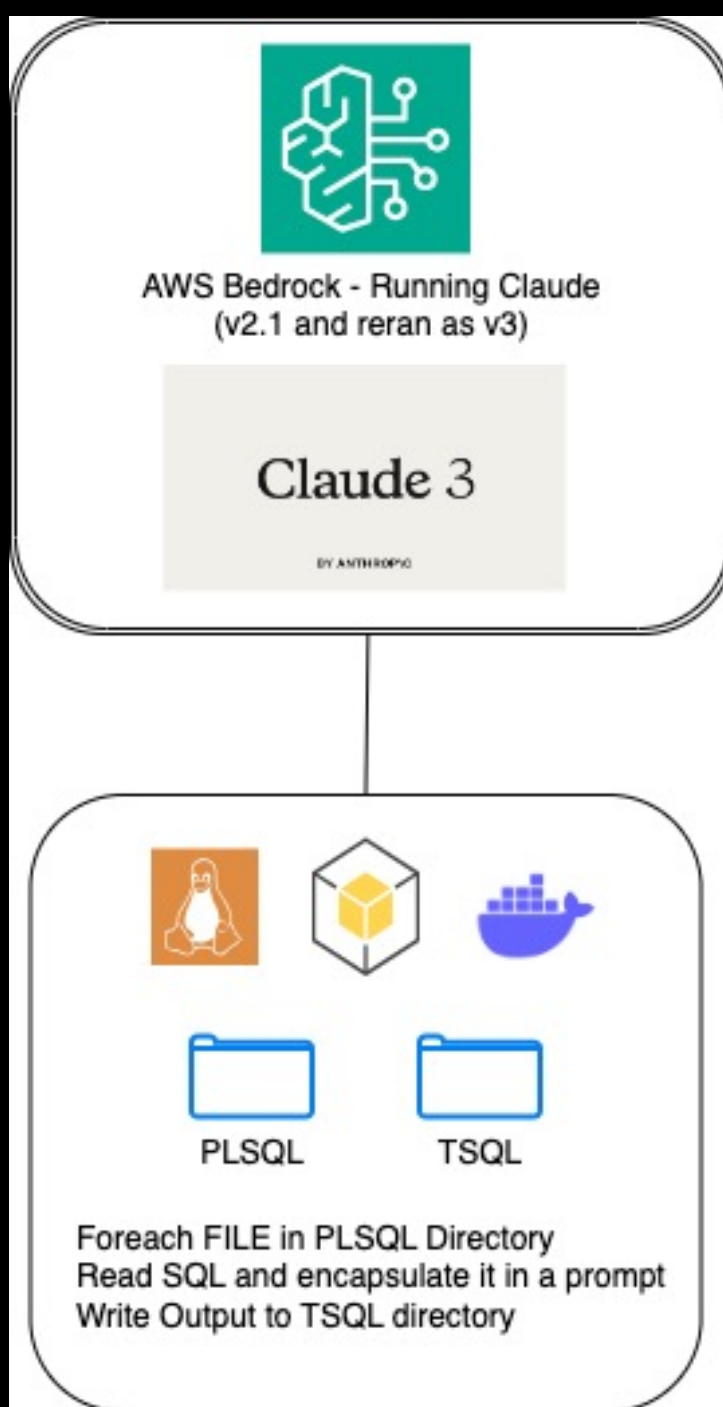
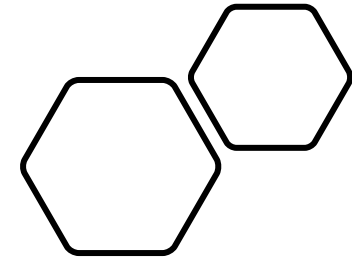
- Prompt templates
- Precision and Clarity
- Contextualization
- Experimentation
- Read the examples and documentation



```
[root@6cdedb76ef19 SQL]# wc -l PLSQL/* | sort -r | head -10
58183 total
12523 PLSQL/GEN_ADT_PKG_BODY.sql
12522 PLSQL/GEN_COVID_VACCINATION_PKG.SQL
2935 PLSQL/X_UCD_PA_SUM_MONTHLY_HURON_SP.SQL.sql
2759 PLSQL/LAB_2268 - Micro Corr Results.sql
2072 PLSQL/LAB_2614 - AP Specific Protocol Turnaround Time.sql
2054 PLSQL/LAB_2250 - ASCP NPQR.sql
2054 PLSQL/LAB_018 - Micro Corrected Results.sql
1964 PLSQL/LAB_2281 - Urinalysis and Culture.SQL
1871 PLSQL/Report_Query13.sql
```

#### Our Opportunity:

- Database vendor change for large EMR application.
- Thousands of reports written using PL/SQL need to be converted to TSQL.
- Fifty reports used for testing



High level overview

```
[root@6cdedb76ef19 SQL]# cat qPrompt
Human: Convert the provided PL/SQL query/code to TSQL for microsoft sql server.

You are generating code.
ALWAYS Keep comments minimal.
ALWAYS make comments escaped for SQL i.e. "--"
ALWAYS encapsulate the generated TSQL code in "-- claudeTSQL --" tags.
NEVER use the tag "-- claudeTSQL --" in any other place.

Check your work carefully to make sure there are no mistakes, errors, or inconsistencies. If there are errors, generate a new version with those errors fixed.
Here is the PL/SQL query/code to convert :

[root@6cdedb76ef19 SQL]# █
```

Human: Convert the provided PL/SQL query/code to TSQL for microsoft sql server.

You are generating code.

ALWAYS Keep comments minimal.

ALWAYS make comments escaped for SQL i.e. "--"











ALWAYS encapsulate the generated TSQL code in "-- claudeTSQL --" tags.

NEVER use the tag "-- claudeTSQL --" in any other place.

Check your work carefully to make sure there are no mistakes, errors, or inconsistencies. If there are errors, generate a new version with those errors fixed.

Here is the PL/SQL query/code to convert :



	Claude	v2	Anthropic	Text	100k	Anthropic's highly capable model across a wide range ...
	Claude 3 Haiku	v1	Anthropic	Text & visi...	200k	Claude 3 Haiku is Anthropic's fastest, most compact m...
	Claude 3 Sonnet	v1	Anthropic	Text & visi...	200k	Claude 3 Sonnet by Anthropic strikes the ideal balance...
	Claude Instant	v1.2	Anthropic	Text	100k	A fast, affordable yet still very capable model, which c...
	Command	v14.7	Cohere	Text	4000	Command is Cohere's flagship text generation model. ...
	Command Light	v14.7	Cohere	Text	4000	Cohere's Command-Light is a generative model that re...
	Embed English	v3	Cohere	Embedding	512	Embed translates text into numerical vectors that mo...
	Embed Multilingual	v3	Cohere	Embedding	512	Embed translates text into numerical vectors that mo...
	Jurassic-2 Mid	v1	AI21 Labs	Text	8191	Jurassic-2 Mid is less powerful than Ultra, yet carefully...
	Jurassic-2 Ultra	v1	AI21 Labs	Text	8191	Jurassic-2 Ultra is AI21's most powerful model for co...

## Why AWS Bedrock?

- Choice of foundation models
- Security and compliance
- Customization
- Serverless
- Ease of use

```

import sys
import boto3
import json
import asyncio
import logging

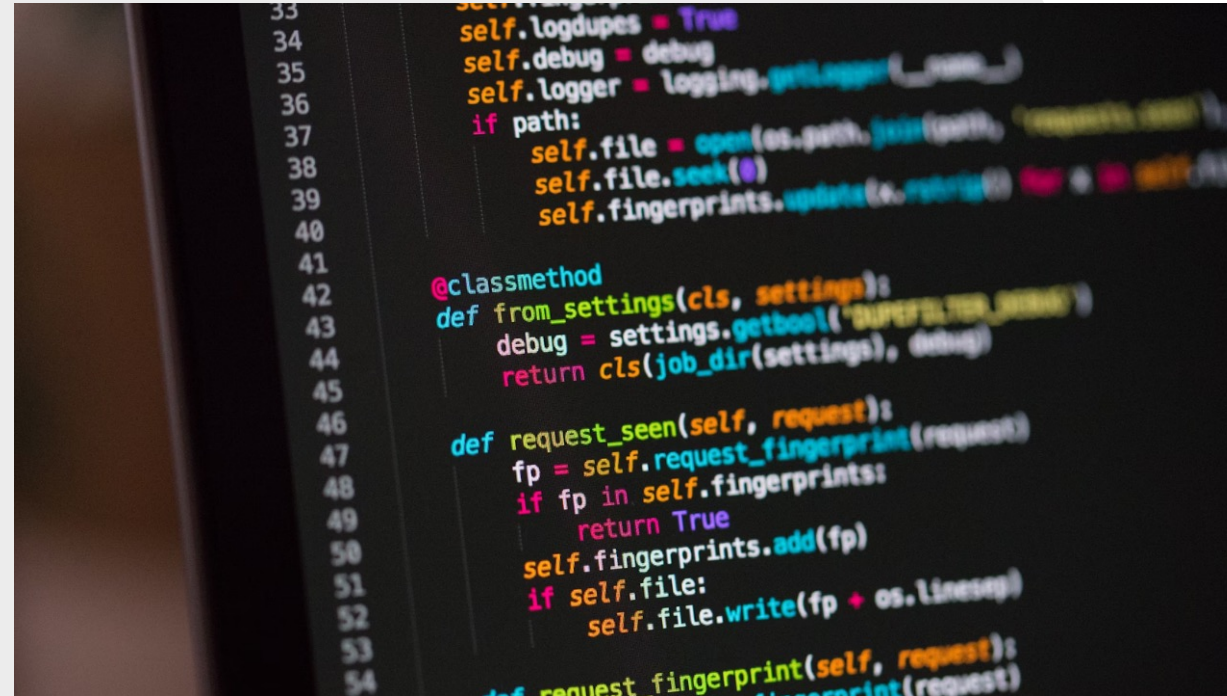
from botocore.config import Config
from botocore.exceptions import ClientError

class ModelInvoker:
    def __init__(self):
        self.config = Config(
            region_name='us-west-2',
            read_timeout=1200,
            connect_timeout=1200,
            tcp_keepalive=True,
            retries={"max_attempts": 10}
        )

        self.bedrock_runtime_client = boto3.client('bedrock-
runtime', config=self.config)

    async def generate_message(self, model_id,
system_prompt, messages):
        """
        Generates a message using the Anthropic Claude model
        with the given parameters.

```



&lt;&lt;&lt;&lt;

**This code was generated by AI. Sadly, as I didn't know I would be using it in a presentation, the prompt file to generate it has been lost. I did ask ChatGPT to generate a prompt from the code, but the results reminded me of the limitations.**

```
#!/bin/bash
```

```
# Directory where the .sql files are located
```

```
sqlDirectory="PLSQL"
```

```
# Local file to concatenate with .sql files
```

```
qPromptFile="qPrompt"
```

```
# Process each .sql file found in the directory
```

```
find "$sqlDirectory" -type f -iname "*.sql" | while IFS= read -r sqlFile; do
```

```
    # Replace 'PLSQL' with 'TSQL' in the directory path
```

```
    newDirPath="${sqlFile/PLSQL/TSQL}"
```

```
    newDirPath=$(dirname "$newDirPath")
```

```
    # Make sure the directory exists
```

```
    mkdir -p "$newDirPath"
```

```
    # Get the base name of the file
```

```
    newFileName=$(basename "$sqlFile")
```

```
    # Set the new file path
```

```
    newFilePath="$newDirPath/$newFileName"
```

```
    # Check if the output file already exists, skip if it does
```

```
    if [ -f "$newFilePath" ]; then
```

```
        echo "Skipping existing file: $newFilePath"
```

```
        continue
```

```
    fi
```

```
    # Concatenate qPrompt with the SQL file and write to argFile
```

```
    cat "$qPromptFile" "$sqlFile" > argFile
```

```
    # Ensure argFile is in UTF-8 (and remove silly ^M)
```

```
    iconv -t UTF-8 argFile -o argFile.utf8
```

```
    sed -e "s/^M//g" argFile.utf8 > argFile.utf8.noM
```

```
    mv argFile.utf8.noM argFile
```

```
    rm argFile.utf8
```

```
    # Run python3 on argFile and redirect output
```

```
    echo "Converting $sqlFile"
```

```
    python3 qClaudeRS.py argFile > "$newFilePath"
```

```
    echo >> "$newFilePath"
```

```
    # Count the occurrences of '-- claudeTSQL --'
```

```
    count=$(grep -c -- '-- claudeTSQL --' "$newFilePath")
```



```
# Check if there are at least 2 occurrences
```

```
if [ "$count" -ge 2 ]; then
```

```
    # If there are at least two instances, apply awk
```

```
    # Find the line numbers for the first and second occurrences
```

```
    first_line=$(grep -n -- '-- claudeTSQL --' "$newFilePath" | head -1 | cut -d: -f1)
```

```
    second_line=$(grep -n -- '-- claudeTSQL --' "$newFilePath" | head -2 | tail -1 | cut -d: -f1)
```

```
    # Edit the file in place
```

```
    # Comment out all lines before the first instance and after the second instance
```

```
    awk -v first="$first_line" -v second="$second_line" 'NR < first || NR >
```

```
        second {print "--" $0; next} {print $0}' "$newFilePath" > tmpfile && mv tmpfile "$newFilePath"
```

```
else
```

```
    # If there are not at least two instances, treat as an error
```

```
    echo "Error: The file does not contain at least two instances of '-- claudeTSQL --'"
```

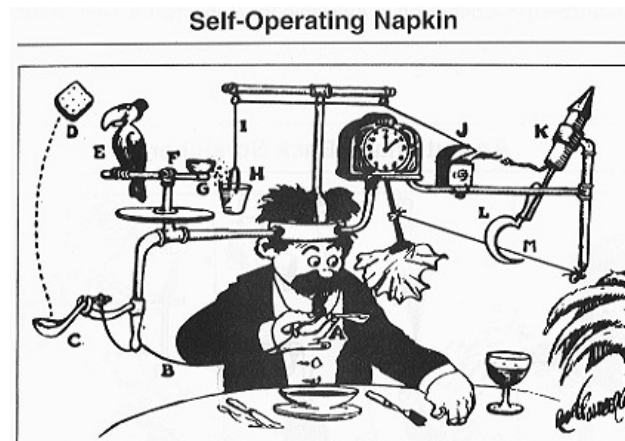
```
    mv "$newFilePath" "${newFilePath}.err"
```

```
fi
```

```
# Take a little break to help avoid throttlingException
```

```
sleep 45
```

```
done
```



AI in the wild

## Quick and dirty shell to prototype

```
# Directory where the .sql files are located
```

```
# Local file to concatenate with .sql files
```

```
# Process each .sql file found in the directory
```

```
    # Replace 'PLSQL' with 'TSQL' in the directory path
```

```
    # Make sure the directory exists
```

```
    # Get the base name of the file
```

```
    # Set the new file path
```

```
    # Check if the output file already exists, skip if it does
```

```
    # Concatenate qPrompt with the SQL file and write to argFile
```

```
    # Ensure argFile is in UTF-8 (and remove silly ^M)
```

```
    # Run python3 on argFile and redirect output
```

```
    # Count the occurrences of '-- claudeTSQL --'
```

```
    # Check if there are at least 2 occurrences
```

```
        # If there are at least two instances, apply awk
```

```
        # Find the line numbers for the first and second occurrences
```

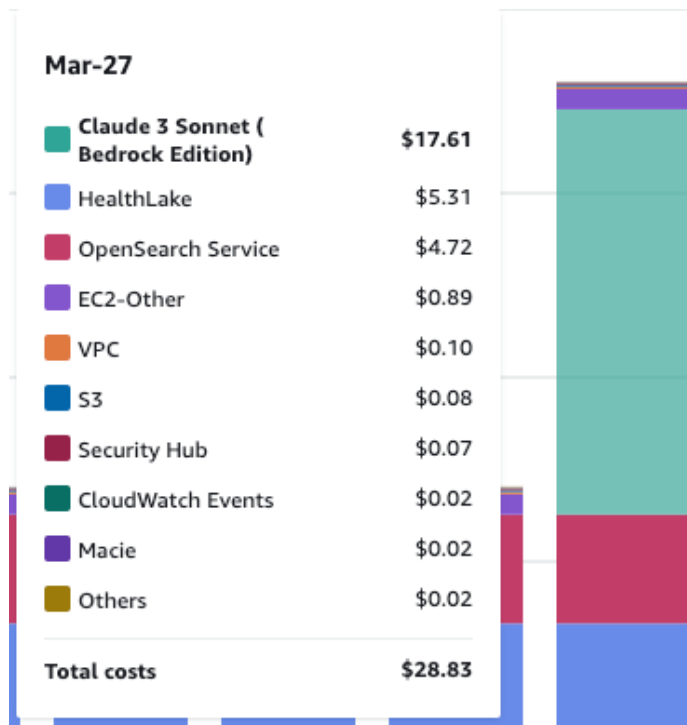
```
        # Edit the file in place
```

```
        # Comment out all lines before the first instance and after the second instance
```

```
        # If there are not at least two instances, treat as an error
```

```
# Take a little break to help avoid throttlingException
```





“generate an image of an AI robot lighting a cigar with burning money”

## Time and Money

- I took longer crafting this presentation than I did prototyping the use of Claude for translating SQL.
- `max_tokens_to_sample`
- The costs to run Claude and translate the 50 files (a few times) was ~18\$ (~8k for all ~3k files)

## PL/SQL

```

SELECT DISTINCT
  a.claim_id,
  a.date_received,
  a.serv_date,
  g.pat_name,
  g.birth_date,
  d.member_num,
  i.prov_name,
  h.vendor_name,
  j.ltrs_generated_date AS Letters_Generated_Date,
  b.pos_type_c,
  CASE e.eob_code_id
    WHEN 419 THEN 'OOA'
    WHEN 202 THEN 'HPRES'
    WHEN 359 THEN 'EMERG'
  END AS Denial_Code
FROM
  ap_claim a
  JOIN ap_claim_px b ON a.claim_id = b.claim_id
  JOIN clarity_eap c ON b.proc_id = c.proc_id

```

## TSQL

```

select distinct
  a.claim_id,
  a.date_received,
  a.serv_date,
  g.pat_name,
  g.birth_date,
  d.member_num,
  i.prov_name,
  h.vendor_name,
  j.ltrs_generated_date as Letters_Generated_Date,
  b.pos_type_c,
  decode(e.eob_code_id,419,'OOA',202,'HPRES',359,'EMERG') Denial_Code
from ap_claim a, ap_claim_px b, clarity_eap c, v_ap_claim_coverage d, ap_claim_
where a.claim_id = b.claim_id
and b.proc_id = c.proc_id
and a.claim_id = d.claim_id
and a.claim_id = e.claim_id
and b.tx_id = f.tx_id
and a.pat_id = g.pat_id
and a.claim_id = h.claim_id

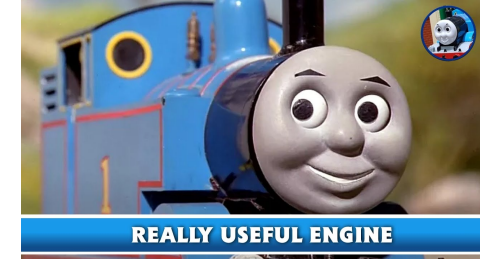
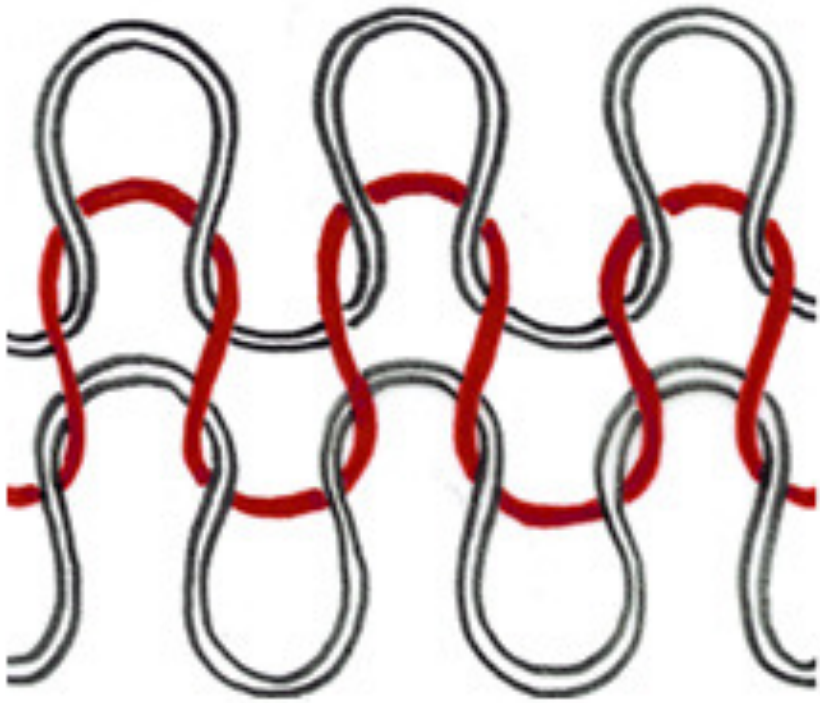
```



## Results?

- The reports team does not have an environment to test the output yet.
- Visual inspection did not reveal any deficiencies.





- **Focus on Collaboration:** Look for opportunities to collaborate with other teams. This way, you can share your expertise while respecting ownership and established processes.
- **Offer Support:** Instead of taking over, can you offer support or resources to the team already working on the issue? This can demonstrate your willingness to help without interloping.
- **Focus on Your Knowledge:** Are there areas that could benefit from sharpening your skills? We are becoming generalist.



“Nostalgia ain't what it used to be.”

— Peter De Vries