# Caderno Bootcamp de Engenharia Dados – SoulCode

# Marcos Valente (github.com/MarcosVCS)

SUMÁRIO

[*Caderno Bootcamp de Engenharia Dados – SoulCode* 1](#_Toc122437990)

[*Marcos Valente (github.com/MarcosVCS)* 1](#_Toc122437991)

[AULA 1 4](#_Toc122437992)

[O QUE FAZ UM ENGENHEIRO DE DADOS? 5](#_Toc122437993)

[AULA 2 5](#_Toc122437994)

[REDES 5](#_Toc122437995)

[AULA 3 6](#_Toc122437996)

[REDES: MÁSCARA 7](#_Toc122437997)

[SERVIDORES 9](#_Toc122437998)

[AULA 4 10](#_Toc122437999)

[COMPUTAÇÃO EM NUVEM 10](#_Toc122438000)

[AULA 5 13](#_Toc122438001)

[GOOGLE CLOUD PLATFORM 13](#_Toc122438002)

[AULA 5 14](#_Toc122438003)

[COMANDOS LINUX BÁSICOS: 15](#_Toc122438004)

[SERVIDORES WEB 17](#_Toc122438005)

[BANCO DE DADOS 18](#_Toc122438006)

[ALTA DISPONIBILIDADE 18](#_Toc122438007)

[CRIAÇÃO DE MODELOS DE VMs 19](#_Toc122438008)

[EDITANDO HTML NO TERMINAL DA VM 20](#_Toc122438009)

[AULA 6 20](#_Toc122438010)

[INSTANCE GROUPS 20](#_Toc122438011)

[LOAD BALANCING 20](#_Toc122438012)

[AULA 7 21](#_Toc122438013)

[SEGURANÇA DA INFORMAÇÃO 21](#_Toc122438014)

[PERMISSÕES A NÍVEL DE PROJETO 24](#_Toc122438015)

[AULA 8 24](#_Toc122438016)

[CLOUD STORAGE 24](#_Toc122438017)

[AULA 9 25](#_Toc122438018)

[EXERCÍCIO PRÁTICO 25](#_Toc122438019)

[CONTAINERS 26](#_Toc122438020)

[KUBERNETES 27](#_Toc122438021)

[AULA 11 27](#_Toc122438022)

[REDES EM CLOUD 27](#_Toc122438023)

[FIREWALL 30](#_Toc122438024)

[VPC NETWORK PEERING (CONEXÃO ENTRE VPCs) 31](#_Toc122438025)

[AULA 12 32](#_Toc122438026)

[CLOUD FUNCTIONS 32](#_Toc122438027)

[MÓDULO PYTHON 32](#_Toc122438028)

[MÓDULO PANDAS 32](#_Toc122438029)

[PRÉ-ANÁLISE 33](#_Toc122438030)

[TRATAMENTO DE DADOS 38](#_Toc122438031)

[VISUALIZAÇÃO DO DATAFRAME 45](#_Toc122438032)

[FILTROS 50](#_Toc122438033)

[GROUP BY 55](#_Toc122438034)

[PLOTAGEM 56](#_Toc122438035)

[VALIDAÇÃO DE DADOS 58](#_Toc122438036)

[RELACIONANDO TABELAS COM PANDAS 60](#_Toc122438037)

[EXPORTAÇÃO DE DATAFRAMES/TABELAS (LOAD) 60](#_Toc122438038)

[MÓDULO BANCO DE DADOS 61](#_Toc122438039)

[MODELAGEM DO BANCO DE DADOS 61](#_Toc122438040)

[COMANDOS MySQL 62](#_Toc122438041)

[CRIAÇÃO DO BANCO DE DADOS NA GCP 62](#_Toc122438042)

[MÓDULO MongoDB (NoSQL) 63](#_Toc122438043)

[COMENTÁRIOS SOBRE NoSQL 63](#_Toc122438044)

[ESTRUTURA MongoDB 64](#_Toc122438045)

[CONEXÃO PYTHON – MongoDB 65](#_Toc122438046)

[CRIAÇÃO DE DOCUMENTO 65](#_Toc122438047)

[VISUALIZAÇÃO DE DOCUMENTOS 66](#_Toc122438048)

[FILTRAGEM 66](#_Toc122438049)

[ATUALIZAÇÃO (UPDATE) 68](#_Toc122438050)

[APAGAR OU DELETAR 69](#_Toc122438051)

[INTEGRAÇÃO COM PANDAS 69](#_Toc122438052)

[ENVIO DO DATAFRAME PARA MongoDB 70](#_Toc122438053)

[DATALAKE E DATAWAREHOUSE 70](#_Toc122438054)

[PROCESSO DE ETL EM UM DATA LAKE 71](#_Toc122438055)

[ESTRUTURAÇÃO DA BASE DE DADOS EM DATALAKE (NA PRÁTICA) 71](#_Toc122438056)

[MÓDULO PYSPARK – AULA 1 71](#_Toc122438057)

[PYSPARK NA PRÁTICA 72](#_Toc122438058)

[MÓDULO PYSPARK – AULA 2 78](#_Toc122438059)

[FILTROS 78](#_Toc122438060)

[TRANSFORMAÇÃO DE COLUNAS 80](#_Toc122438061)

[VISUALIZAÇÃO DE DADOS 81](#_Toc122438062)

[TRANSFORMAÇÃO DOS DADOS 85](#_Toc122438063)

[MÓDULO PYSPARK – AULA 3 89](#_Toc122438064)

[VISUALIZAÇÃO DE DADOS 89](#_Toc122438065)

[MANIPULAR DATAS (DATE) 89](#_Toc122438066)

[SALVAR DATAFRAME (WRITE – SAVE) 91](#_Toc122438067)

[CONEXÃO DE COLUNAS (JOINS) 92](#_Toc122438068)

[MÓDULO SPARK – AULA 4 95](#_Toc122438069)

[CONEXÃO COM GOOGLE CLOUD STORAGE (DATA LAKE) 95](#_Toc122438070)

[CONVERTER DATAFRAMES ENTRE SPARK E PANDAS 97](#_Toc122438071)

[MÓDULO SPARK – AULA 5 97](#_Toc122438072)

[TIPOS BÁSICOS DE DADOS DO SPARK 97](#_Toc122438073)

[STRUCT TYPES 98](#_Toc122438074)

[WINDOW FUNCTIONS 99](#_Toc122438075)

[SPARK SQL 103](#_Toc122438076)

[CONECTANDO MYSQL AO PYTHON 104](#_Toc122438077)

[LOOKER STUDIO (ANTIGO DATA STUDIO) 107](#_Toc122438078)

[LOOKER STUDIO NA PRÁTICA 107](#_Toc122438079)

[DATAPROC 108](#_Toc122438080)

[CONFIGURANDO DATAPROC 108](#_Toc122438081)

[PIPELINES 109](#_Toc122438082)

[APACHE BEAM NA PRÁTICA 110](#_Toc122438083)

# AULA 1

## O QUE FAZ UM ENGENHEIRO DE DADOS?

* Tratamento de dados:
  + Nem sempre o dado que o engenheiro obtém para a empresa está correto. Exemplos:
    - nem sempre está no tipo correto (números podem ser apresentados em string também)
    - mesmos dados são apresentados de maneiras diferentes (ex: pedágio, pedagio, pedágios...)
      * Necessário fazer um tratamento de dados, nesse caso (normalizar dados, variáveis, objetos...)
* Migração de database:
  + Deixar dados que sejam legados disponíveis e atualizados
* Disponibilizar dados para que analistas de dados possam traçar métricas e decisões de negócio
* Em resumo: ETL (extração, transformação e carregamento)

Interface gráfica do usuário, Site

Descrição gerada automaticamente

# AULA 2

## REDES

* Quatro características atribuídas a dispositivos com MAC
  + IP: localização lógica de um dispositivo dentro de uma rede
    - Dispositivos se comunicam por IP
    - Geralmente é atribuído de forma aleatória (roteador atribui um IP a um dispositivo na rede)
    - Por meio do IP, faz-se parte de uma rede, permitindo a troca de informações
    - Utilizado dentro da rede para trafegar informações (não se utiliza o MAC para trafegar informações)
    - Atualmente há duas versões de IP:
      * IPv4
        + X.X.X.X (quatro octetos – sendo que cada octeto pode ir de 0 a 255)
        + Desde 2014, não há mais IPs disponíveis para se vender
      * IPv6
        + X.X.X.X.X.X (seis octetos – sendo que cada octeto vai de 0 até F)
        + A máscara deixará de existir quando o IPv6 assumir todos os IPs
  + Máscara
    - Resolve o problema de escassez de IPv4
    - Antes de haver a máscara, todas as pessoas recebiam IPs públicos diretamente na máquina (todos pertenciam à mesma rede, que é a internet)
    - Quando se criou a máscara, passou a haver o IP público (ou válido) e o IP privado (ou não válido)
      * IPs privados são distribuídos dentro de uma rede local (atribuídos pelo roteador)
        + Não são entendíveis para a internet
    - O roteador possui no mínimo dois MACs
      * Um MAC recebe um IP público (enviado pela provedora de internet)
      * O outro recebe o IP privado da própria rede
    - O dispositivo da rede local, ao acessar a internet, apodera-se do IP público do roteador por meio da máscara
    - A máscara, ao solucionar o problema da escassez de IPs, também permitiu delimitar redes
  + Gateway
    - Determina as pontes entre as redes
      * Possibilita a saída de dados de uma rede para outra
    - Por onde a informação entre as redes trafega
    - Na grande maioria dos casos, o roteador é o próprio gateway
  + DNS
    - Tradutor entre ser humano e máquina
    - Traduz domínios e IPs

# AULA 3

## REDES: MÁSCARA

* Criação de redes menores, internas, isoladas, onde se repete o IP público, ou válido (IPv4)
* Faixas de IP privado não serão utilizadas no IP público, mas poderão ser repetidas em redes novamente
  + Ex: 192.168.0.1 (primeiro IP da rede)
* Faixas IPs privados (não podem ser públicos):
  + 10.0.0.0 – 10.255.255.255
  + 172.16.0.0 – 172.16.255.255
  + 192.168.0.0 – 192.168.255.255
  + Observação: IPs com final 0 são utilizados somente para designar a rede, não sendo atribuído a nenhum aparelho (desenho de topologia)
* Faixa 169.254 é reservada para loop-back (pedido de acesso a IP privado foi negado, e esse IP foi atribuído a ele para que não ficasse sem configuração – no entanto, não acessa a internet)
* Primeiro IP da rede privada é reservado ao roteador (ex: 192.168.0.1), o último é reservado para broadcast (ex: 192.168.0.255), transmitindo informações para toda a rede
* Máscara é um número (a rede possui uma única máscara – configurada no roteador – o DHCP distribui a máscara da rede, os DNS e os IPs privados)
  + No Brasil, a máscara padrão costuma ser 255.255.255.0
  + Trata-se de um endereço de máscara aberta
* Máscara divide a rede interna
* IPs privados podem ser distribuídas para até 253 dispositivos (255 menos o roteador e menos o broadcast)
* O número da máscara delimita o número de IPs privados utilizados em uma rede
  + *A conta é: 255 menos o número do último octeto da máscara menos 2 (IP do roteador e IP do broadcast)*
    - Último octeto da máscara serve para limitação
  + Exemplo máscara 255.255.255.100: será possível atribuir 153 IPs privados
  + Ao definir uma máscara para a rede, define-se o número de dispositivos que podem se conectar a ela (questão de segurança)
  + Rede aberta de IPs é aquela que possui máscara 255.255.255.0
    - No entanto, é possível aumentar o número de IPs privados em uma rede
    - Para isso, é necessário modificar os outros octetos da máscara (sobretudo o terceiro)

**Aumentando número de IPs privados disponíveis em uma rede**

* Ao definir o terceiro octeto de uma máscara, aumenta o número de sub-redes
  + Cada sub-rede pode ter até 253 IPs privados
* Exemplo: uma rede com máscara 255.255.**254**.0 possui duas sub-redes
  + Nesse caso, será possível atribuir 253\*2 IPs nessa rede
* Exemplo 2: uma rede com máscara 255.255.**252**.0 possui quatro sub-redes
  + Repare que a máscara opera por meio de subtrações (tanto limitando o número de IPs, quanto aumentando o número de sub-redes)
* Os números de IP privado serão ampliados pois aumentará, no intervalo de IPs disponíveis, o número do terceiro octeto dos IPs
  + Exemplo: se houver uma rede sem sub-redes, o intervalo de IPs privados poderá ser 192.168.0.0 até 192.168.0.255
    - Caso se crie uma sub-rede, o intervalo aumentará para 192.168.0.0 até 192.168.**1**.255

**Em resumo...**

* Para limitar o número de IPs em uma rede, sempre se configura o último octeto da máscara
* Para aumentar o número de IPs em uma rede, configura-se novas sub-redes, atribuindo novos números ao terceiro octeto da máscara

**CIDR**

* Forma resumida de apresentar a máscara
  + Maior CIDR é 32 (totalmente fechada)
  + CIDR 24 é totalmente aberta (sem haver sub-rede)
  + Menor é 0

**Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente**

**Tabela

Descrição gerada automaticamente**

## SERVIDORES

* Clientes acessam servidores em busca de serviços ou recursos
* Arquitetura cliente-servidor (baseado em requisições e retornos)
  + O Youtube, por exemplo, porta-se como um servidor de mídia (provedor de serviço)
  + Cliente (navegador) solicita ao Youtube acesso a um vídeo (cliente solicita serviço para servidor; servidor envia serviço de volta)
    - É feito download do vídeo pelo navegador
  + Em suma, o servidor retorna ao cliente (nesse caso, o navegador) as informações demandadas (por meio de download)
    - Por isso, não se tem acesso diretamente ao servidor

**Tipos de servidor**

* Servidor de aplicação
  + Servidor utilizado para executar aplicações corporativas (geralmente sistemas gerenciais de uma empresa) – sistemas internos
  + Pode ser sistema desktop, mobile...
* Servidor de arquivos
  + Sistemas para armazenas e compartilhar uma grande quantidade de informações entre equipamentos e usuários, mantendo o gerenciamento centralizado
* Servidor de banco de dados
  + Toda aplicação desenvolvida precisa armazenar os dados dessa aplicação
  + Coração da empresa (precisam ser protegidos, possuir redundância, estar alocados em local específico, o banco de dados)
* Servidor WEB (ex: Hostgator)
  + Armazena front-end de aplicações WEB (HTML, CSS e JavaScript)
  + Armazena algumas partes do back-end de uma aplicação WEB
* Servidor de mídia (ex: Youtube, Netflix)
* Servidor de e-mail
* Servidor FTP
  + Permite download e upload de arquivos via conexão com protocolo FTP (File Transfer Protocol)
* Servidor Proxy
  + Funciona como um filtro entre seu roteador e a internet
  + Existem vários servidores proxy entre o roteador e a internet
  + Finalidades:
    - Controle de tráfego (controlar o que entra e o que sai da rede)
      * Serve, por exemplo, para controlar uso de internet de funcionários
    - Servidor cache
      * Melhora a velocidade de acesso a aplicações web (se já foi acessada por outra pessoa)

# AULA 4

## COMPUTAÇÃO EM NUVEM

* Como armazenamento de Big Data demanda um crescente poder de armazenamento, é necessário trabalhar com computação em nuvem, em que é possível facilmente expandir o hardware dos computadores
  + Nuvem pode ser expandida ou diminuída a qualquer momento (recursos são “infinitos”) – por isso diz-se que trabalha “on demand”
  + Escalonamento vertical: melhorar capacidade de uma máquina que já estava em operação na nuvem

Escalonamento horizontal: expandir o número de máquinas, para que trabalhem em prol de um mesmo conjunto

* Contratar operadora de cloud -> montar hardware na nuvem -> subir aplicações/ sistemas/ arquivos -> disponibilizar para público (funcionários ou clientes)
  + Utiliza-se o navegador para isso

Nuvens públicas:

* Operadoras que abrem seus recursos (hardware e software) para que usuários da nuvem possam ter um fácil acesso; mas os dados de todos os usuários da nuvem, enquanto serviço, estão no mesmo espaço (em síntese, são servidores compartilhados)
* Vantagens: custo menor, dado o compartilhamento de hardware; provedor de serviços fornece manutenção; escalabilidade praticamente ilimitada; alta confiabilidade (ampla rede de servidores)

Nuvem privada:

* Aluguel de todo espaço físico do servidor (serão usadas exclusivamente por toda empresa ou organização)
* Pode estar localizada fisicamente no datacenter local da aplicação ou estar hospedada em um datacenter remoto, terceirizado
  + Havendo um ambiente local, mantém-se a característica de nuvem porque utiliza-se software específico para trabalho a distância
* Vantagens: maior controle e privacidade (hardware exclusivamente seu); possibilidade de montar hardware específico, de acordo com necessidade (o que, dependendo do uso, pode ser impossível na nuvem pública)

Nuvem híbrida:

* Utiliza-se tanto de nuvens públicas quanto privadas

**IaaS , PaaS , SaaS**

Diagrama

Descrição gerada automaticamente

Infrastructure (as a Service) (IaaS)

* Baseado em máquinas virtuais (emulação de um hardware dentro de outro hardware)
  + Montagem de servidores via máquinas virtuais (solicitação de hardware e software) – compete a nós criar as máquinas virtuais, bem como gerenciá-las
  + Responsabilidade sobre o funcionamento da máquina virtual é do usuário
  + Cliente monta ambiente de execução (inclusive responsável por manter aplicações funcionando)
* Provedor de serviços em nuvem gerencia a infraestrutura para você — servidores, rede, virtualização e armazenamento de dados — via internet
  + Papel da Google Cloud: manter o hardware operacional
* O usuário pode acessá-la por meio de uma API ou painel de controle e, essencialmente, aluga a infraestrutura

Platform (as a Service) (PaaS)

* Serviço para desenvolvedores de sistemas
  + Cuidado com funcionamento da aplicação (dados e acesso)
  + Exemplo: contratação de serviço de hospedagem de aplicação WEB
* Não há criação de máquina virtual, tampouco de gerenciamento de hardware
  + Compra de servidores já padronizados
  + Não é necessário instalar ambiente computacional (Java, Python, MySQL...)

Serverless (Function as a Service - FaaS)

* Cloud fica responsável por administrar o hardware necessário para prover a aplicação (capaz de expandir ou diminuir, em tempo real, dependendo da demanda por armazenamento e processamento)
* Desenvolvedor cuida somente da aplicação

Software (as a Service) (SaaS)

* Nuvem é utilizada somente pela aplicação
  + Usuário somente consome o serviço
* Exemplos: Google Docs, Google Drive, Netflix...

# AULA 5

## GOOGLE CLOUD PLATFORM

* Obrigatoriamente, cada conta no Google Cloud deve ter um projeto
* Projetos podem ser transferidor para outros usuários
  + Para além das pessoas ligadas a ele
  + Em outras palavras, é possível transferir a posse de projetos
* Questões financeiras, dentro do GCP, são computados por projeto
* O número do projeto e o ID do projeto devem ser únicos
* Todo projeto possui um dono
* O front-end padrão do Google Cloud está hospedado em uma máquina virtual vinculada a seu projeto
  + Trata-se do console
* O shell é o terminal da mesma máquina
  + O shell possui 10GB de espaço por padrão
* Tudo que é criado dentro do GCP, é criado dentro do projeto
* Qualquer produto pode ser criado tanto no console quanto no shell
  + Escolher o que deixa mais à vontade
  + No entanto, convém mencionar que, com a experiência, o uso do shell é mais rápido
  + Em nosso curso, somente um produto é configurado pelo shell
  + Observação: produtos do GCP em teste só podem ser acessados pelo shell

**Organização da Cloud**

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

* Organizações conseguem criar pastas (importante para nível organizacional)
* Projeto é o conjunto de recursos e serviços (ambiente de infraestrutura) dentro da google cloud

# AULA 5

* Indicação de S.O. para montagem de máquina virtual: Debian
  + Menor para ocupar espaço (maior velocidade de processamento)
  + Desenvolvedores Debian prezam por estabilidade e segurança (down-side: algumas aplicações, por deverem ser testadas para segurança, podem ser ligeiramente defasadas)
* Tentar utilizar o mesmo SO das outras máquinas, para conexão em rede entre elas funcionar melhor
* HD equilibrado é o suficiente para o dia a dia
  + Especialmente quando máquina não é utilizada para processar dados
  + Sistema Operacional, depois de atualizado, não chega a consumir 1GB do disco
* Cada máquina possui seu próprio Firewall

## COMANDOS LINUX BÁSICOS:

* Observação importante: Linux não dá confirmação

“apt update” : buscar atualizações (procurar de acordo com fontes próprias, inclusive de aplicações)

“apt upgrade” : pega o que tem de atualizável, baixa e instala

“apt install” : instalar aplicações específicas

* No entanto, é necessário ser administrador da máquina para instalar ou modificar aplicações
* Para se portar como administrador (super usuário):
  + Comando “sudo su” o torna super usuario (perigoso, pois continua como super usuário até que se diga o contrário (comando “exit”))
  + Utilizar “sudo” à frente do comando.
    - Utiliza-se temporariamente o direito de super usuário
    - Exemplo: “sudo apt update”

“pwd” : saber onde cursor está (print work directory)

“ls” : monstra o que há dentro do diretório em que o usuário está (list)

“ls – l” : mostra de maneira mais detalhada

* Mostra permissões, dono, momento em que foi criada, etc.

“clear” : limpa tela

“mkdir” : criar nova pasta (make directory)

* Inserir nome em seguida
  + Exemplo: “mkdir aula”
  + Exemplo criando três pastas ao mesmo tempo “mkdir aula aula1 aula2”
  + Para conferir, utilizar comando “ls”

“cd” : adentrar em novo diretório (change directory)

* Se estiver no mesmo diretório, basta botar o nome da pasta
* Se não estiver no mesmo diretório, deve-se passar o nome absoluto dela (caminho)

“cd ..” : voltar para nível acima

Abrir editor de texto: há dois editores no terminal do Linux:

* “vim”
  + Comando “vim *nome\_arquivotexto*.txt” cria arquivo de texto
    - Se o arquivo de texto já existir, esse comando abre ele no vim
  + Abre arquivo novo automaticamente
  + Alguns sistemas demandam que se clique a tecla “INSERT” ou “i” antes de digitar (senão ficará no modo de visualização)
  + Tecla “ESC” volta para modo de visualização
  + Salvar arquivo: ir até final do arquivo e digitar “:” para receber novo comando (pode ser comando de salvar, sair, deletar...)
    - Salvar: “:w” (write)
    - Sair: “:q” (quit)
    - Sair sem salvar “:q!”
    - Salvar e sair: “:wq”
      * Não saia antes de salvar!
* “nano”

“rm *nome\_arquivo*” : remove arquivos

“rm – d *nome\_pasta*” : remove pastas, se vazia

“rm – d – r *nome\_pasta*”: remove pastas e arquivos dentro

“rmdir *nome\_pasta*” : remove pastas, se vazia

* Comando legado, uma vez que o “rm” faz tudo

“mv /null” : envia para o vazio (move to null)

* Excluído sem rastros

Texto

Descrição gerada automaticamente

## SERVIDORES WEB

* Serviços para desenvolvedores
* Otimizado para aplicações web
* Provê aos usuários a página solicitada, via HTML, CSS e Javascript
* Existe software específico para servidor
  + Apache
  + NGINX

Uma imagem contendo Diagrama

Descrição gerada automaticamente

* Protocolo HTTP (faz requisição baseada em hipertexto)
* O Apache organiza o código fonte, liberando suas funcionalidades para o usuário à medida que forem requisitados
  + Apache enquanto software servidor, responsável por responder a requisição HTTP

## BANCO DE DADOS

* Dado é tudo que pode ser registrado
* Informação é um dado que possui algum valor para alguma pessoa/organização

Dois principais tipos de bancos de dados informatizados:

* Relacionais (SQL)
* Não relacionais (NO SQL)
  + Evolução dos dados (ex: dados de redes sociais – dados muito grandes)
    - Bancos relacionais não eram preparados para trabalhar com uma quantidade de dados tão grande
  + Existem diferentes tipos de bancos não relacionais
  + MongoDB

## ALTA DISPONIBILIDADE

* Serviços precisam ter alta disponibilidade (estarem constantemente disponíveis)
* Disponibilidade é importante para extração de dados
* Se o número de requisições for maior do que o processador do servidor conseguir entregar, o serviço poderá cair
* Não existe alta disponibilidade se houver uma única máquina como servidor web (no mínimo duas máquinas, para garantir a redundância)

Auto Scaling:

* Escalonamento automático
  + Disponibilidade de recursos da nuvem de acordo com a demanda
  + Define-se um mínimo de máquinas trabalhando como servidor (ex: duas máquinas) e um máximo (definido por quem contrata o serviço - o número máximo na GCP é mil)
  + Garante alta disponibilidade

Auto Healing

* Verificação automática das aplicações rodando na VM, para ver se estão rodando bem
* Se for constatado problema na aplicação, a máquina parará de prover o serviço, devendo ser criada uma nova para substituí-la
* Configurações básicas:
  + Protocolo HTTP (busca na raiz do servidor web)
  + Porta 80
  + Request path (arquivo, nesse caso html, que será detectado): index.html
  + Logs: não (pois são cobrados)

Load Balance:

* Balanceador de carga
* Equipamento de rede que recebe as requisições e as distribui aos servidores (usuários acessam o load balance e não as VMs específicas)
  + Organiza o tráfego de requisições (para as diferentes VMs no cluster de máquinas)
  + Administra o auto healing (o load balance é o primeiro a perceber o problema em uma máquina virtual) – sendo que a requisição vai ser redirecionada para outra
* Essa load balance, na cloud, será uma VM (em um servidor físico, costuma ser um equipamento de rede)
* Load balance deve ter endereço fixo (endereço da aplicação)

## CRIAÇÃO DE MODELOS DE VMs

* Útil para tornar o auto scaling e auto healing possíveis
* Facilita criação rápida de máquinas virtuais padronizadas

## EDITANDO HTML NO TERMINAL DA VM

Texto

Descrição gerada automaticamente

* Caminho /var/www/html é formado pelo Apache
* Inserindo esse código, o usuário é transferido para o vim, para lá editar o HTML

# AULA 6

## INSTANCE GROUPS

* Passos para criação de instâncias com autoscalling e autohealing (instance groups)
  + Criação da Instance Template
  + Criação do protocolo de Health Check
  + Criação do Instance Group

## LOAD BALANCING

* Load balancing é responsável por controlar o tráfego, direcionando as requisições entre as máquinas dos instance groups
* Os clientes não acessam diretamente as máquinas, mas sim a load balancing (que possui IP externo próprio e fixo)
  + Controla o tráfego, instituindo uma fila de clientes
* Dividido em duas partes
  + Configuração entre cliente e load balance (front-end do load balance)
    - Configura-se o IP, o protocolo e a porta
    - IP efêmero será fixo enquanto existir o Load Balance
  + Configuração entre load balance e instâncias (back-end do load balance)

Configurar Load Balancing

* Configurar da internet para as máquinas virtuais (controla tráfego da internet (clientes) para VMs)
* Configurar um load balance global
  + Configurar frontend
  + Configurar backend
    - A princípio, nosso backend será de instance group
    - Protocolo HTTP
    - Timeout: quanto tempo espera para considerar uma requisição uma falha
    - Apontar instance group em que deverá operar
    - Apontar porta utilizada no instance group
    - Modo de balanceamento determina como o load balancer vai direcionar as requisições entre as VMs
      * Por utilização (baseia-se no uso da CPU de uma VM)
      * Por rate (requests por segundo)
      * Para aplicações que utilizam mto processamento, recomenda-se optar por utilização/ aplicações que terão muitos requests devem ser gerenciadas por rate
    - Observação: é possível montar inúmeros backends (em regiões e zonas diferentes)
      * Se um backend estiver sobrecarregado, pode-se direcionar o tráfego por meio de outro

# AULA 7

## SEGURANÇA DA INFORMAÇÃO

* Engenheiros de dados lidam com algumas partes do tema segurança
* Tema dinâmico: sempre mudando (com base em novas ameaças, novas demandas...)
* Deve-se analisar quantas camadas de segurança são necessárias a uma determinada segurança
  + Necessário inserir no mínimo duas camadas de segurança
* Segurança de dados é um tema sensível (“dados são o novo petróleo”)
  + Dado sensível é todo aquele que pode dar um prejuízo a alguém

Ataques comuns:

* Golpes de engenharia social:
  + Utilizam-se da manipulação para persuadir e roubar informações privadas
  + Geralmente vinculado ao estelionato
  + Em um sistema seguro, o elo mais fraco é o ser humano
* Ataque DoS e DDoS (ataques de negação de serviço)
  + Ataque com múltiplas requisições (vazias) ao servidor ao mesmo tempo, para exceder a capacidade de entrega dos serviços (derrubando o serviço)
  + Acontecia mais quando não havia nuvem escalável
* Ransomware
  + Extorsão e “sequestro” de informações, exigindo-se um resgate para liberá-los
  + Vírus do tipo worm, cuja finalidade é capturar arquivos de um computador/sistema e criptografá-los

VPN

* Primeiro passo de segurança, ao adentrar uma rede, é o Firewall
* O VPN faz com que computadores remotos pertençam à mesma rede
  + Quando isso acontece, o Firewall não protege mais (pois ele trabalha ao conectar-se de uma rede externa para uma rede interna)
  + O que é perigoso, pois máquinas externas estavam desprotegidas. O VPN fazia com que os perigos associados a um computador particular se transmitissem diretamente para redes internas (sobretudo de empresas)

Pilares da segurança da informação

Uma imagem contendo Texto

Descrição gerada automaticamente

* Confidencialidade
  + Tornar algo que pode ser sigiloso confidencial
  + Mesmo em posse dos dados, uma pessoa não autorizada não os consegue ler nem interpretar (é aplicada técnica de confidencialidade em cima do documento)
  + Técnica empregada: esteganografia
    - Exemplo de programa que faz isso: camoflage
* Integridade
  + Garantia de que o arquivo, mesmo depois de passado por várias mãos, se mantém fiel ao original
    - Há crimes especializados na mudança de dados
  + Tecnologia empregada: Cálculo de Hash
    - Cálculo de bits, para saber se o arquivo é idêntico
* Disponibilidade
  + Garantir o acesso aos dados
  + É função do engenheiro de dados programar a disponibilidade dos dados
* Autenticidade
  + Validar a autorização do usuário para acessar, transmitir e receber determinados dados
  + Confirmação da identidade do usuário
    - Mecanismo básico: login e senha
* Irretratabilidade (Não retratação/ “Não repudio”)
  + Autenticação entre máquinas (permite comunicação entre máquinas)
  + Tecnologia empregada: Certificados digitais
    - Certificados digitais são variáveis
    - Servidor confirma o certificado por meio de um cálculo

Segurança na GCP

Gráfico, Gráfico de barras

Descrição gerada automaticamente

* Relembrando:
  + IaaS = Infrastructure as a Service
  + PaaS = Platform as a Service
  + SaaS = Software as a Service

## PERMISSÕES A NÍVEL DE PROJETO

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

# AULA 8

## CLOUD STORAGE

* Ferramenta da GCP voltada para armazenamento de objetos (todos formatos possíveis que podem ser armazenados)
  + Inclusive objetos que serão disponibilizados em uma página web
  + Também para formação de Data Lakes
* Necessário designar um local para armazenar os objetos
  + Esse local chama-se “bucket”
  + Mais avançado do que um simples diretório: o bucket pode ser identificável em qualquer lugar do mundo, por meio da GCP
    - Existem buckets públicas (acesso livre, cujos dados podem ser compartilhados com quaisquer pessoas)
* Classes de armazenamento
  + Objetos são salvos de acordo com a classe de armazenamento
  + Aqueles objetos que são utilizados diariamente são postos em uma classe específica (custos próprios)
  + Objetos que são usados com menos frequência são postos em outras classes (valor menor de armazenamento)
    - Há uma gradação nesse sentido (existem quatro classes principais):
      * Standard (dados são utilizados sempre, com certa frequência)
      * Nearline (dados que ficam armazenados por pelo menos 30 dias – programa-se que serão utilizados em algum momento depois desse período)
        + Assim como nas outras categorias, se forem utilizados antes desse “período de carência”, será cobrada uma taxa superior
      * Coldline (dados que ficam armazenados por pelo menos 90 dias, sem serem utilizados)
        + Observação: dados em buckets nearline, se não forem utilizados em 90 dias, transformam-se automaticamente em “coldline”, para que o usuário seja tarifado menos
      * Archive (dados que ficam armazenados por pelo menos 365 dias, sem serem utilizados)

Interface gráfica do usuário, Aplicativo, Teams

Descrição gerada automaticamente

# AULA 9

## EXERCÍCIO PRÁTICO

Crie um template de instâncias para máquina virtual com as seguintes configurações:

- Configuração da Máquina: N1 - F1-micro

- Linux Debian 10

- Permitir Tráfego HTTP e HTTPS

- Colocar script de automação / inicialização conforme documentação

Em seguida criar um grupo de instâncias com as seguintes características:

- Região / Zona aleatória

- Auto Scaling: Mín: 2 , Máx: 4, CPU: 70%

- Auto Healing: Verificar o arquivo index da aplicação, configurar conforme os parâmetros vistos em aula

Criar uma bucket com as seguintes configurações

- Single Region

- Access control: Fine-grained

- Class: Standard

- Protection: None

- Criar uma pasta chamada img na bucket

- Colocar 3 imagens aleatórias na pasta img

- Tornar as 3 imagens públicas

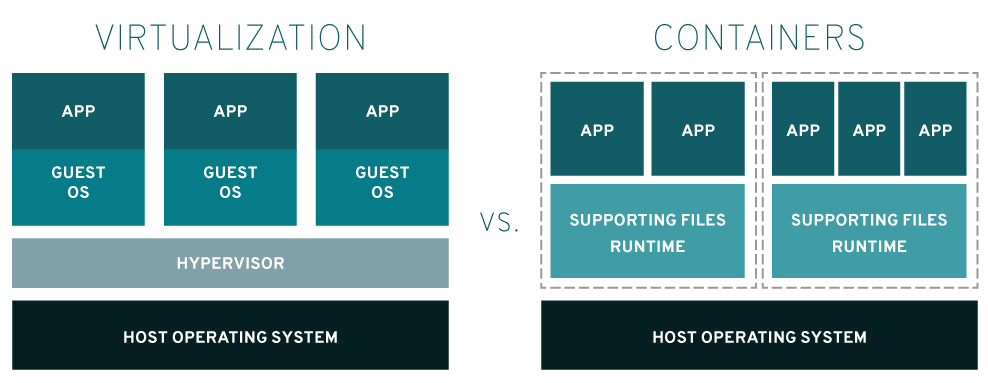
Em seguida alterar o arquivo index.html dentro das máquinas e no script de automação substituir as tags <p></p> e o seu conteudo pela seguinte tag <img src="caminho\_da\_imagem">, onde caminho\_da\_imagem será a URL da imagem pública na bucket que foi criada (Escolha uma imagem qualquer)

- Tente colocar também as outras duas imagens juntas na mesma página

No final configue o Load Balance conforme orientação durante as aulas e veja se está tudo funcionando conforme o esperado!

CONTAINERS

* Diferentemente das VMs, que virtualizam máquinas virtuais (necessitando cada uma um SO), o container virtualiza sistemas
  + Nos containers, a aplicação não está atrelada a um SO
  + Não consome memória ram para manter o sistema operacional
  + Velocidade de implementação muito mais rápida



* Para isso, utiliza-se as instruções do sistema operacional do próprio servidor para sustentar as aplicações
  + Instruções compartilhadas entre os diferentes usuários do servidor
* Vantagens:
  + Velocidade de implementação muito mais rápida
  + Menor consumo de espaço no HD
  + Menor consumo de memória (pois não precisa sustentar SO)
* Ainda se utilizam VMs porque existem aplicações que ainda não podem ser rodadas em containers (ex: banco de dados relacionais)
  + Também, por questões de legislação, alguns serviços não podem contar com bibliotecas compartilhadas
* Containers são muito voláteis
  + Não são feitos para guardar dados
    - Se precisar, relaciona-se com um cloud-storage
  + São feitos para processar informações (microserviços)
  + São desligados rapidamente
    - Para poder ser ligados muitas aplicações no mesmo sistema operacional

## KUBERNETES

* Orquestrador de containers
  + Verifica pedidos, filas, destruição de containers
  + Atua entre os containers e os clientes

# AULA 11

## REDES EM CLOUD

VPN (Virtual private network)

* VPN conecta o usuário a uma rede doméstica diferente, fazendo com que a rede mundial entenda que você está em outro local
* Empresas ou locais que possuem seus servidores ou computadores podem proteger seus dados para usuários de fora da rede
  + A única forma de acessar esses dados é fazer parte da mesma rede desse servidor (conexão via LAN, evitando o firewall externo da rede)
  + Caso esteja remoto, pode se utilizar uma VPN (WAN)
* VPN cria um “túnel” conectando o usuário à rede da empresa]
* O computador do usuário terá dois IPs, um da casa (vinculado ao provedor) e outro da empresa
* Explicação:
  + Toda rede que se conecte à internet possuirá um IP público e um IP privado (atribuído pelo roteador)
  + É atribuído ao usuário um IP da rede da empresa (IP público e IP privado)
  + A requisição sai do computador do usuário, passa pelo roteador da empresa e depois passa para a rede mundial
    - Isso faz com que a rede fique mais lenta
    - Contudo, o usuário pode acessar os arquivos como se estivesse naquele local
    - O usuário submete-se às regras de segurança da rede que adentrou

Virtual Private Cloud (VPC)

* Criação de uma nuvem privada virtual dentro de uma nuvem privada
* Criação de uma rede própria dentro do GCP
* Importante para proteger os recursos dentro da GCP, pois dentro da mesma rede os riscos se propagam
* Permite setorizar máquinas virtuais (mesmo que dentro do mesmo projeto e da mesma zona)
  + Determina com quem as máquinas conversarão
* Aumento da segurança de forma simples e gratuita
  + Criação de uma rede em que você possui o controle e a segurança

Criação de VPC

* Necessário criar duas VPCs (VPC-a e VPC-b)
  + Cada rede possuirá um IP diferente
* Configurar VPC
  + Enquanto todos não estiverem com IPv6, utilizar IPv4
  + Subnets customizado
    - Nomear a subnet (isso porque dentro de uma VPC podem haver várias subnets – para diversas regiões)
    - Determinar região
    - Utilizar IPv4, pois ainda não há IPv6 para todos
    - Em IP range, deve-se determinar o IP da rede e o número de dispositivos que podem se conectar a ela (IP e máscara)
      * Exemplo: 192.168.5.0/24
      * Nesse caso, podem ser alocados 253 dispositivos nessa rede (ver aula sobre máscara)
      * Trata-se de um IP interno, não precisando se preocupar com sua disponibilidade
    - Private Google Access (determinar que somente pessoas permitidas pelo IAM acessem a rede)
      * Deixar OFF, pois de outra forma não poderia servir como servidor WEB
    - Flow logs
      * Deixar em OFF, a princípio, pois trata-se de um serviço pago
      * Nada impede que no futuro seja ativado
  + Firewall rules
    - A Google oferece soluções padrão, mas nós criaremos nossas próprias
    - Se nenhuma opção for escolhida, determina-se que tudo que “entra” é barrado e tudo que “sai” é liberado
  + Modo de roteamento dinâmico
    - Selecional global (permite que qualquer pessoa no mundo encontre a VPC, podendo se conectar a ela por meio de VPN)
      * Se escolhesse regional, dificultaria o acesso a pessoas de outras regiões
  + Criar VPC
* Montando rede (passos)
  + Criar modelo de instância (instâncias serão vinculadas à VPC – passo não obrigatório)
    - Habilitar firewall interno da máquina
    - Nas opções avançadas, configurar networking
      * Configurar networking interfaces
      * Substituir default pela VPC e sub-rede desejadas
      * Não configurar outras opções de networking
    - Introduzir script de inicialização (para instalar APACHE 2 e criar site simples)
  + Criar grupo de instâncias
    - Observação: só será possível criar se o instance group estiver na mesma região da TPC vinculada ao modelo de instância
    - As VMs terão o IP interno determinado pela VPC

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* Nesse momento, o firewall impede acesso aos dados da máquina (servidor WEB e terminal SSH)
  + Necessário configurar firewall

## FIREWALL

* Firewall controla o fluxo de informações
* Firewall controla o acesso às portas de rede (permite ou bloqueia portas – não existe meio termo)
  + Não é antivírus, não é filtro de dados
* Existem dois protocolos ou portas em que se solicitam aplicações web: HTTP (porta 80), HTTPS (porta 443)
  + O pedido da máquina até o servidor passa por essas portas
  + A resposta (arquivos recebidos) passam por outras portas, chamadas portas altas (são mais de 2 mil portas)
    - Isso possibilita acessar mais de uma aplicação web ao mesmo tempo
* Portas existem em qualquer dispositivo de rede (roteadores, placas de rede, etc.)
  + Todas as portas são virtuais
  + Por isso que o próprio sistema operacional (bem como os antivírus) possuem Firewall
* O correto é possuir o Firewall da rede, Firewall da VPC e Firewall da máquina
  + Compete ao arquiteto de cloud determinar os níveis de proteção

Configurando Firewall

* Abrir configurações da VPC
  + Adicionar regras de firewall
  + Nomear firewall com função (ex: “libera-ssh”)
  + Logs não são obrigatórios (há custo e serão gerados muitos logs)
  + Prioridade (prioridade da regra)
    - A instrução do Firewall que tiver mais prioridade será executada, em caso de instruções contraditórias
    - OBS: a prioridade é 0 (quanto mais perto de 0, maior a prioridade)
  + Escolher direção do tráfico (ingress = entrada; egress = saída)
    - Exemplo: para acessar o SSH da máquina virtual, é necessário entrar nela
  + Action on match: permitir ou negar acessar porta
  + Targets:
    - All instances in the network (todos que estão dentro da rede serão contemplados pela regra)
    - Outros casos serão utilizados quando se necessitar maior segurança
  + Source Filter (filtro de fonte) – possível adicionar mais de um:
    - Determina quais IPs podem acessar essa porta
    - 0.0.0.0/0 permite que qualquer IPv4 acesse essa porta
      * Boas práticas: somente aplicar esse IP para acesso às portas 80 e 443 (responder pedido de sites)
  + Protocols and ports
    - Specified protocols and ports
    - Observação: existem portas que possuem protocolo e número, que possuem somente protocolo (ex: icmp) e que possuem somente número
    - TCP: determinar porta (ex: porta do protocolo SSH é 22)
    - Other: determinar protocolo (ex: SSH)

**“Regra de incêndio/ terremoto”**: sempre criar uma regra de firewall com prioridade 65355 que impede qualquer acesso (se em algum dia houver ataque, transformar (editar) essa regra em prioridade 0)

* Targets: all instances in the network
* Não permitir acesso a nenhuma porta

## VPC NETWORK PEERING (CONEXÃO ENTRE VPCs)

* Permitir a conexão entre VPCs (informações indo e voltando)
  + VPCs e dispositivos vinculados passarão a se conhecer
  + Necessário configurar duas vezes (uma para cada origem)
* Aplicação prática: permitir que grupo de instâncias que possuem mais liberações (regras de segurança) se comuniquem com outra rede que possua outras liberações
  + Exemplo: servidores web comunicando-se com bancos de dados

Configuração (lembrando que se deve configurar duas vezes):

* Entrar em VPC network peering
* Criar conexão (A -> B)
  + Escolher VPC de origem
  + Escolher VPC do projeto ou fora do projeto (destino)
  + Escolher VPC de destino
  + Assinalar somente “Export subnet routes with public IP”
* Criar conexão (B -> A) seguindo mesmos passos

# AULA 12

## CLOUD FUNCTIONS

* Em resumo: Gatilho de execução para códigos (por isso, é mais utilizado para desenvolvedores)
* Funções em nuvem (FaaS)
  + Derivação do SaaS
  + Cobrado não por segundo, mas por execução
  + Grande parte dessas funções opera sobre contêineres
* Pequenos contêineres com códigos já prontos, que possuem finalidades específicas

# MÓDULO PYTHON

* Como já tenho um caderno sobre essa linguagem de programação, não farei anotações por aqui...
* Vocês podem consultá-lo neste link: [CADERNO\_PYTHON.docx](https://1drv.ms/w/s!AhTFc97VQheyi-FvBzLNGjrdFV4JsQ?e=w4OtXO)

# MÓDULO PANDAS

* Biblioteca em Python para trabalho com dados (atualmente, é utilizado para trabalhar com bases pequenas)
* Uma das limitações da biblioteca é o volume de dados (para bases maiores, será necessário usar PySpark)
  + Mas existe uma vantagem para o Pandas: não é necessário operar em cluster (além de ser visualmente mais amigável)

**Fases do ETL**

* Extração
  + Arquivo consolidado
    - Observação: o arquivo original deve sempre estar disponível
  + Dado extraído
* Transformação
  + Limpeza de inconsistências (normalização)
  + Verificação de registros duplicados
  + Verificação dos tipos de registros
  + Filtros, inserções, etc.
* Carregamento

## PRÉ-ANÁLISE

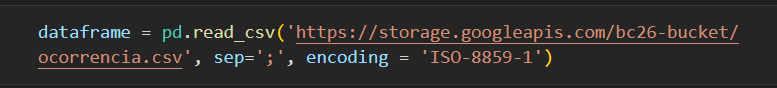
**Importando Pandas**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

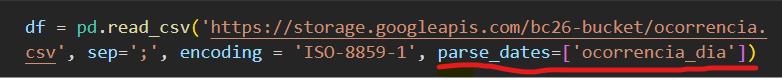
**Importando base de dados (“Data frame”)**

* + Necessário atribuir uma base de dados a uma variável, doravante chamada de dataframe (df)



* + Esse comando abre o arquivo, o lê e o transporta para a memória RAM (não modifica a base de dados original)
  + No método “.read\_*extensão*” é essencial apontar tipo de arquivo
  + Entre parêntesis está uma string com o caminho da base de dados
    - Se o arquivo estivesse no computador, deveria ser passado o endereço interno
  + O comando “sep =” indica os separadores de dados, os quais nos arquivos CSV costumam ser vírgulas (se nenhum separador for indicado, o Pandas entende que a base utiliza vírgulas)
  + Enconding diz respeito à codificação da base de dados
    - Pandas é multicodificação (consegue ler diferentes codificações)
    - Se nenhuma codificação for atribuída, opta-se por padrão por UTF-8
    - Se não souber, utilizar a biblioteca chardet
  + Obs: preferir utilizar a variável “df” no lugar de “dataframe”

**Definir tipos das colunas ao importar dados (especialmente colunas que contenham data)**

****

* Comando “parse\_*tipo* = [‘*nome\_coluna*’]” é utilizado para modificar o tipo dos dados da coluna no momento da apresentação
* Nesse caso, modificou-se uma coluna com datas, para que seu tipo seja de datas, seguindo padrão internacional (AAAA,MM,DD)

**Realizar backup do dataframe**

* Fazer um backup, antes de começar a normalizar os dados, é essencial

Interface gráfica do usuário

Descrição gerada automaticamente

* Fazer backup de uma coluna somente

Texto

Descrição gerada automaticamente

* + Esse comando criará uma nova coluna na tabela, que servirá como backup (no caso, ‘cidade\_backup’)
  + Depois, se quiser eliminar a coluna, basta eliminá-la

Tela preta com letras brancas

Descrição gerada automaticamente

**Configurando opções do Pandas**

* Comandos para determinar máximo de colunas e linhas que serão exibidas

**Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente**

**Exibindo dataframe**

* Para exibir o dataframe, basta chamar a variável

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

* Método “.head()” exibe primeiras linhas
  + Deve ser passado o número de linhas (se não for, por padrão exibem-se 5)

**Tela preta com letras brancas

Descrição gerada automaticamente**

* Método “.df.tail()” exibe últimas linhas
  + Deve ser passado o número de linhas (se não for, por padrão exibem-se 5)

Tela de computador com letras e números em fundo preto

Descrição gerada automaticamente

* Mostrar tipos de dados contidos em colunas

Texto

Descrição gerada automaticamente

* + Observação: quando uma coluna apresenta tipos misturados (i.e. colunas contendo strings e números), o Pandas entende que ela contém “objetos”

**Verificação do número de linhas e colunas no dataframe**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Verificação mais completa das colunas (nomes, tipos de dados e quantidade de entradas (não nulos))**

Texto

Descrição gerada automaticamente

## TRATAMENTO DE DADOS

**Tornar coluna index**

* Index determinado pelo Pandas deixará de existir
* Importante para localizar logs ou ids mais rapidamente
* Para tornar os valores de uma coluna index, esses devem ser únicos
  + É possível perguntar para o Pandas se os valores são únicos

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

* Transformando a coluna em novo rótulo

Interface gráfica do usuário

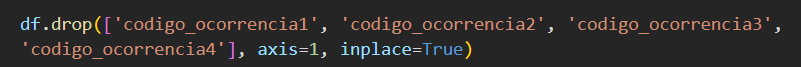
Descrição gerada automaticamente com confiança média

* Resetando o rótulo



* + Obs: por esse método, a tabela perde sua ordenação

**Dropar colunas (eliminar)**

****

* Repare que, no comando, é passada uma lista contendo as colunas que se pretende dropar
* Após a lista, é designado o eixo em que será dropado (se coluna ou linha)
  + Pode-se utilizar “columns” ou “rows” , bem como 1 (colunas) ou 0 (linhas)
* Comando “inplace” designa se alteração deve ou não ser aplicada ao dataframe
  + Por padrão está como False (resultado será somente um preview da tabela)

**Renomear colunas**

* Para isso, utiliza-se o método “.rename(columns={‘nome\_coluna\_modificada’: ‘novo\_nome’})”
* Atenção: esse comando pode ser utilizado para renomear linhas
* Preferir nomes de coluna em caixa baixa e sem caracteres especiais (também não deve haver espaço vazio)

**Texto

Descrição gerada automaticamente**

**Concatenar colunas**

* Ambas as colunas devem apresentar dados do mesmo tipo
* No exemplo, iremos concatenar as colunas ‘dia’(DateTime) e ‘horario’(string)
  + Necessário, primeiro, converter dados da coluna ‘dia’ para string

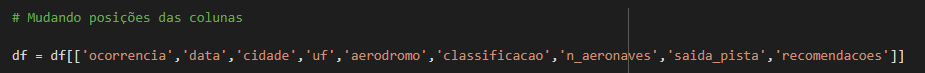
Interface gráfica do usuário

Descrição gerada automaticamente

* Repare que foi criada uma nova coluna (‘data’)
  + As colunas originais não foram deletadas

**Mudar posição das colunas**

* Necessário esquematizar novamente todo o dataframe (transforma-se o dataframe inteiro, criando um novo ou sobrescrevendo o original)



* + Repare que é necessário passar uma lista com o nome das colunas dentro de outra lista (comando do Pandas)

**Verificar se, em uma coluna, existem valores repetidos (ou se são todos únicos)**

**Interface gráfica do usuário, Texto

Descrição gerada automaticamente com confiança média**

**Uma imagem contendo Texto

Descrição gerada automaticamente**

**Código para perceber strings em coluna onde deveriam constar somente números (mesmo que no formato string)**

**Texto

Descrição gerada automaticamente**

**Limpar dados**

* Para saber quais são os elementos únicos dentro de uma coluna, utiliza-se o método “pd.unique(*nome\_dataframe*[‘*nome\_coluna*’])”
  + Pandas analisa todas as tuplas e retorna uma lista com todos os elementos

**Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente**

* + Ordenando consulta alfabeticamente:

**Texto

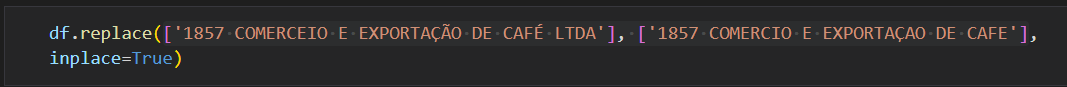
Descrição gerada automaticamente**

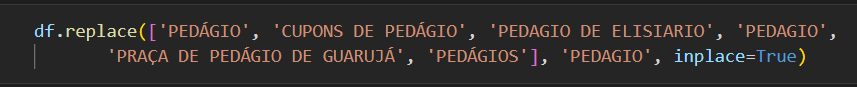
* Para anular inconsistências, utiliza-se o método “.replace([‘valor1’, valor2, ...] , pd.NA)

**Interface gráfica do usuário, Texto

Descrição gerada automaticamente com confiança média**

* + Nesse caso, os campos com inconsistências passarão a conter valores nulos (NA do Pandas)
* Para alterar valores contidos em um dataframe, utilizar o método “.replace[‘valor1’, valor2, ...] , [‘novo\_valor’])
  + Frisando: primeira lista contém valores que serão modificados, enquanto a segunda, o valor substituto (somente um)



****

* Método replace em cima de colunas específicas:

****

* + Repare que após a seleção do dataframe, é delimitado a coluna por um ponto
* Para fazer correções específicas em dataframes, utiliza-se o método “df.loc[]”

**Texto

Descrição gerada automaticamente**

* + Nesse caso, o campo ‘cidade’ (que é uma coluna), na linha 0, passará a ser “SANTOS”
* Também é possível utilizar filtros para corrigir todos as incorrências em uma coluna
  + Primeiro, cria-se um filtro, para selecionar todas as incorrências erradas

**Texto

Descrição gerada automaticamente**

* + - Repare que o nome Santos foi grafado equivocadamente, com um “m”
    - O filtro seleciona todas as incorrências “SAMTOS” dentro da coluna “localizacao” dentro do dataframe “df”
  + Depois, atribui-se um novo valor a essas incorrências

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

* + - Repare que é passado uma lista com o filtro e a coluna que será alterada
    - Como se está atribuindo diretamente um valor para o objeto (por meio de um sinal de igual), não é necessário confirmar a modificação com “inplace = True”

**Corrigir dígitos específicos dentro de uma string**



* Argumento “regex” permite com que se corrija dígitos dentro de uma string (se não utilizasse esse atributo, a substituição só ocorreria se houvesse uma string exatamente igual a “O”)
* Nesse caso, trocou-se o caractere “O” por “0” (zero), erro comum em dataframes

**Transformar dado em nulo, sem mudar tipo da coluna**

* Utilizar NaN (da biblioteca NumPy)



* + Repare que np.NaN é um método da biblioteca NumPy

**Mudar tipo de coluna**



* Nesse caso, mudou-se do tipo Object para Float64
* Observação: método só funciona se os valores forem compatíveis

## VISUALIZAÇÃO DO DATAFRAME

**Mostrar colunas específicas de um dataframe**

**Tela de computador com fundo preto

Descrição gerada automaticamente**

* Repare que é necessário criar uma lista sobre uma lista

**Selecionar células completas**

* Para consultar células inteiras, utilizar o método “.loc[*número\_célula*]”

Texto

Descrição gerada automaticamente

* + Repare que todas as colunas são retornadas
* Para localizar um intervalo de células:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* + Novamente, todas as colunas são retornadas

**Selecionar todos os valores de uma coluna específica**

Texto

Descrição gerada automaticamente

**Visualizar maior e menor valor dentro de uma coluna (que contenha números)**

* Maior valor

**Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente**

* Menor valor

**Interface gráfica do usuário

Descrição gerada automaticamente com confiança média**

**Visualizar média dos valores dentro de uma coluna (que contenha números)**

Texto, Site

Descrição gerada automaticamente com confiança média

**Visualizar dados numéricos de toda a tabela**

Tela de computador

Descrição gerada automaticamente

**Visualizar dados numéricos de uma coluna**

Texto

Descrição gerada automaticamente com confiança média

**Consultar quantas células possuem valores nulos (NA – Not Available)**

**Uma imagem contendo Tabela

Descrição gerada automaticamente**

**Consultar número de inserções nas colunas (mesmo que inconsistentes)**

Uma imagem contendo Tabela

Descrição gerada automaticamente

* Repare que são retornados os números de valores que contém cada coluna
* Em outras palavras, foram deduzidos os valores nulos

## FILTROS

* Para filtrar uma tabela, é necessário criar um filtro e localizar a partir desse filtro

**Filtro que seleciona valores vazios (NA, NULL, NaN, NaT, etc.)**

**Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente**

* Repare que a sintaxe do filtro é “*dataframe*.*coluna*.isna()”

**Criando novo dataframe a partir de um filtro**

**Tela de computador com letras e números em fundo preto

Descrição gerada automaticamente**

* Nesse caso, criou-se um novo dataframe contendo somente os registros em que o “UF” corresponde a “SP”

**Filtros compostos**

* É possível filtrar uma tabela por mais de um critério
* Para isso, devem ser criados diferentes filtros, os quais deverão ser chamados pelo parâmetro *nome\_dataframe*.loc[*filtro1* & *filtro2* & *filtro3...]*

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

* + Nesse caso, serão apresentados os registros de incidentes ocorridos em São Paulo em 2021
* Filtrar por um critério ou por outro
  + Sintaxe: *nome\_dataframe*.*coluna*.**isin([*lista\_critérios*])**

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

* + Nesse caso, foram filtradas todas as ocorrências que aconteceram em estados da região Sul

**Filtrar a partir de datas**

* Deve haver coluna formatada para DateTime (dt)

Texto

Descrição gerada automaticamente

* + Observação: nesse caso, a coluna “dia” possui as informações de data (ali estão presentes o dia, mês e ano da ocorrência

**Filtrar um dataframe e apresentar colunas específicas**

**Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente**

## GROUP BY

* Método “.groupby()” agrupa todos os resultados de uma coluna, podendo ser empregados outros métodos para análise do dataframe
* Exemplo:

Uma imagem contendo Aplicativo

Descrição gerada automaticamente

* + Nesse caso, agruparam-se os valores de uma coluna, e o Pandas retornou o número de vezes em que esses valores foram localizados
* Por default, a classificação é por ordem alfabética dos valores
* Para classificar por valores, basta usar o método “.sort\_values(ascending=False)”

Texto

Descrição gerada automaticamente com confiança média

* Observação: por padrão, o Group By não apresenta os valores nulos
  + Para exibi-los, deve-se passar um parâmetro específico dentro do método ‘.groupby()’:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Mostrar registros com maior e menor incidência na coluna**

* Mostrando três registros com maior número de incidências:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

* Mostrando três registros com menor número de incidências:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

## PLOTAGEM

* Para plotar os dados, deve-se agrupar os resultados e utilizar, na mesma linha de código, o método “.plot.*tipo\_gráfico*()”

Gráfico, Gráfico de barras

Descrição gerada automaticamente

* + Nesse método, selecionou-se o gráfico de barras
  + Para ver outros tipos de plotagem, consultar a documentação do Pandas:

[pandas.DataFrame.plot — pandas 1.5.2 documentation (pydata.org)](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html)

* Para determinar o tamanho da imagem:

Gráfico, Gráfico de barras

Descrição gerada automaticamente

* Determinar nome dos eixos



## VALIDAÇÃO DE DADOS

* Necessário utilizar a biblioteca Pandera
* Deve-se validar os dados de uma tabela ao menos duas vezes:
  + Após dropar as tabelas que não serão utilizadas, renomear as utilizadas e corrigir erros muito evidentes
  + Antes da última verificação geral

**Criação do Schema**

Texto

Descrição gerada automaticamente

**Chamando Schema**

**Texto

Descrição gerada automaticamente**

* Feito isso, analisar os erros reportados, para saber quais registros contém inconsistências (diferem do que foi pedido na hora de formatar o Regex)
* Exemplo:
  + Repare que ele aponta as células em que há inconsistência

**Texto

Descrição gerada automaticamente**

* Se depois de rodar o schema aparecer o dataframe, não há inconsistência

## RELACIONANDO TABELAS COM PANDAS

* Só é possível relacionar as tabelas que possuírem uma coluna com as mesmas informações (i.e. “id”, “número\_ocorrência”)
  + O nome da coluna deve ser igual também
* Escolher quem será o dataframe de base (da esquerda)
* Realizar o merge



* + Opção ‘outer’ faz com que o Pandas relacione as duas tabelas e considere relações que possam estar faltando
  + Opção ‘inner’ só apresenta células que apresentem relação entre as duas tabelas (descartando informações de uma das tabelas que não se relacione com a outra)

## EXPORTAÇÃO DE DATAFRAMES/TABELAS (LOAD)

* Sempre validar os dados antes de dar um load
* Possível exportar para uma miríade de tipos

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

* + Há mais opções abaixo, como SQL
* Nomeando arquivo



* Observação: importante retirar o rótulo antes de exportar o dataframe
  + Se isso não for feito, o rótulo será exportado como uma coluna



# MÓDULO BANCO DE DADOS

## MODELAGEM DO BANCO DE DADOS

MODELO DER

Texto preto sobre fundo branco

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

* Atributos compostos são aqueles que são compostos por outros atributos
  + Exemplo: endereço é composto por Rua, Número, CEP, Cidade etc.
* Atributos multivalorados são aqueles que podem apresentar mais de um valor
  + Exemplo: é possível que o mesmo usuário possua mais de um número de telefone

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

* Exemplo de lado N: tabela de bebês (N bebês possuem 1 médico)
* Exemplo de lado 1: tabela de médico que realizam partos em bebês (1 médico possui N bebês)

Calendário

Descrição gerada automaticamente

## COMANDOS MySQL

VER MEU CADERNO SOBRE MySQL

<https://1drv.ms/w/s!AhTFc97VQheyi9tM_mq3RWqi516x3Q?e=XPiAcl>

## CRIAÇÃO DO BANCO DE DADOS NA GCP

* Criar servidor SQL no Google Cloud (máquina que servirá como servidor de banco de dados MySQL)
  + Criar nova instância
    - Nomear instância
    - Designar uma senha
    - Versão 8.0 do MySQL
    - Selecionar configuração:
      * Ambiente de produção: funcionará no ambiente da empresa, tendo maior disponibilidade
      * Ambiente de desenvolvimento: voltado para desenvolvimento da aplicação (mais barato)
    - Selecionar região (única zona ou múltiplas zonas)
    - Escolher tipo de máquina, com base no tamanho do banco e no número de querries que receberá
    - Escolher tipo e tamanho do disco
    - Escolher configurações de Data Protection
      * Backup automático
      * Proteção contra eliminação da instância
    - Criar instância
* Acessar a máquina e acessar o MySQL (pelo Cloud Shell)

# MÓDULO MongoDB (NoSQL)

## COMENTÁRIOS SOBRE NoSQL

* NoSQL = Not Only SQL
  + Quando esse tipo de banco de dados é chamado “não relacional”, comete-se um equívoco, pois podem ser traçadas relações dentro dele

Diferenças entre SQL e NoSQL

* SQL:
  + Modelo rígido (a estrutura deve ser criada e respeitada)
    - Dificulta o armazenamento simultâneo de dados
  + Escala vertical (todos os dados precisam estar na mesma estrutura)
  + Investimento na criação da infraestrutura
* NoSQL:
  + Modelo flexível (não há tanta necessidade de seguir modelo)
    - Não exige estrutura rígida, nem obriga respeitar os tipos dos dados
  + Escala horizontal (foi projetado para responder ao “gargalo vertical” do SQL)
  + Investimento na criação da inserção de dados (planejar concepção da inserção e da consulta)
    - Banco não precisa de estrutura definida, já que é flexível
* Existem quatro tipos gerais de bancos NoSQL

Gráfico

Descrição gerada automaticamente

* + Cada tipo apresenta uma vantagem para determinado cenário
  + Todos possuem uma ideologia de escalabilidade horizontal (possível particionar o banco de dados entre várias instâncias, HDs etc.)
    - Isso permite processar os dados em cluster
* MongoDB é o banco NoSQL que tabalha com documentos (em JSON)
  + Gerenciamento de dados por meio de documentos
  + Como os documentos estão separados, a consulta pelos dados pode ser mais rápida (carrega-se somente o que é solicitado)
  + Desvantagem com relação ao SQL: visualização em JSON
    - Para corrigir esse problema, deve-se utilizar Pandas ou PySpark
* Tipo de grafos ainda não é popular, embora seja altamente otimizada
* Tipo chave-valor é utilizado para armazenar sessões
  + Rápido para armazenar pequenos volumes de informação (cliques, sessões, links)
  + Tecnologia: Redis
* Tipo colunar é semelhante ao SQL, mas cada coluna é carregada individualmente (aumenta velocidade de processamento)
  + Tecnologia: Cassandra

## ESTRUTURA MongoDB

* Cada projeto possui um database
* Dentro dos databases, existem coleções (uma ou múltiplas)
* Dentro das coleções, existem documentos (um ou múltiplos)
* Logo, para identificar um dado, é necessário indicar o database, a coleção e o documento

## CONEXÃO PYTHON – MongoDB

* Na plataforma do MongoDB, selecionar “Connect” -> “Connect your Application”
  + Selecionar Python e acesso por chave criptografada
  + Copiar código e colar no Python

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

## CRIAÇÃO DE DOCUMENTO

* Todo comando é antecipado pelo nome da coleção
* Método ‘.insert\_one’ insere somente um documento; ‘.insert\_many’, múltiplos documentos

**Inserindo um documento:**



* Não há limite de parâmetros para criação de um documento
* Preferir popular o banco documento a documento (demanda menos processamento, mais fácil controlar o processo)

**Inserindo múltiplos documentos**

* Cada dicionário corresponderá a um documento (obs: eles devem estar dentro de uma lista)

****

## VISUALIZAÇÃO DE DOCUMENTOS

**Visualização de todos os documentos de uma coleção**

Texto

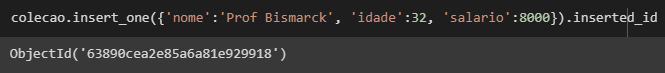
Descrição gerada automaticamente

* Repare que todo documento possui um id
  + O id identifica o documento, para que possa ser referenciado no futuro

**Visualização de ids dos dados**

**Texto

Descrição gerada automaticamente**

****

## FILTRAGEM

* Exemplo 1: buscar registros (documentos) que possuem idade maior que 30

Texto

Descrição gerada automaticamente

* + O “$” dentro do parâmetro, indica um comando de comparação
    - ‘gt’ = greater than
  + Obs: MongoDB é case-sensitive
* Exemplo 2: buscar documentos que possuem idade menor ou igual a 30:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

**Controles especiais para busca**

**Texto

Descrição gerada automaticamente**

**Busca por nome**

**Texto

Descrição gerada automaticamente**

* Obs: a String deve ser obrigatoriamente idêntica àquela cadastrada no BD

**Buscar por múltiplos termos**

**Texto

Descrição gerada automaticamente**

* Repare que é necessário apontar uma lista com os termos, precedidos do caractere especial $

**Pesquisas compostas**

**Texto

Descrição gerada automaticamente**

* Pesquisar com “ou”

**Texto

Descrição gerada automaticamente com confiança média**

* + Nesse exemplo, pesquisa-se quem possui nome “Marcos Valente” ou idade maior que 30 anos

## ATUALIZAÇÃO (UPDATE)

* Atualizar ou adicionar parâmetros dentro dos documentos
  + Se o parâmetro modificado não existir, ele é criado (NoSQL é flexível)
* Orientação: preferir utilizar o parâmetro “.update\_many” do que o “.update\_one”

Texto

Descrição gerada automaticamente

**Excluir uma coluna de um documento**

* Necessário redefinir todos os parâmetros do documento, deixando a coluna em tela de fora

## APAGAR OU DELETAR

* Cautela ao utilizar o parâmetro “.delete\_many”
* Não existe backup para MongoDB

**Apagando documento (registro inteiro)**

****

* Repare que é feita uma busca por quem possui nome “cliente 0”
  + Se fosse utilizado o parâmetro “.delete\_many”, todos os registros que se enquadrassem nesse filtro seriam deletados

**Apagando a coleção (cautela!)**

****

## INTEGRAÇÃO COM PANDAS

* Necessário utilizar as bibliotecas Pandas e pymongo ao mesmo tempo

Texto

Descrição gerada automaticamente

* Considere dar um drop na coluna “\_id”, herdada do MongoDB

## ENVIO DO DATAFRAME PARA MongoDB

* Necessário criar uma nova coleção
* Necessário que a biblioteca numpy esteja importada, para que o MongoDB receba valores NaN

Texto

Descrição gerada automaticamente

# DATALAKE E DATAWAREHOUSE

* Maior diferença entre eles é como os dados estarão organizados dentro deles
* Utilizados para fins diferentes:
* Datalake (MongoDB/ Cloud Storage)
  + Orientação: guardar uma grande quantidade de dados (“infinita”), em grande velocidade
  + Aceita dados sem estrutura definida
  + Dados são trabalhados por engenheiros, não sendo consumidos por analistas
* Datawarehouse (BIgQuery ou “Cluster” de Bancos relacionais)
  + Aceita somente dados estruturados
  + Dados são otimizados
  + Bancos de dados relacionais que possuem informações estruturadas, para dar suporte a um fluxo de dados contínuo
  + Consegue gerir uma grande quantidade de dados, limitado pela solução (consome muito mais recursos financeiros)
  + Analistas podem consumir esses dados

## PROCESSO DE ETL EM UM DATA LAKE

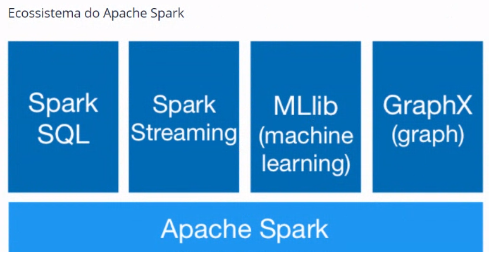
* A ingestão de dados pode ocorrer de duas formas:
  + Batch: base de dados consolidada
  + Streaming: alimentados a todo momento (não é simultâneo, mas acontece com constância)
* Os dados são armazenados no Cloud Storage (e produtos concorrentes)
  + Importante zelar pela segurança dos dados
* O processamento se dá em Apache Spark
* Os dados serão disponibilizados em SQL, NoSQL...
* Os dados serão consumidos por cientistas de dados, analistas de dados, dashboards, aplicações em python...

## ESTRUTURAÇÃO DA BASE DE DADOS EM DATALAKE (NA PRÁTICA)

* Estruturar um database para cada empresa ou solução
* No caso do MongoDB, organizar coleções por laço temporal ou setores da empresa/ solução
  + Mesma lógica deve ser aplicada aos Cloud Storage
* Deve haver, no mínimo, duas organizações: dados brutos e dados tratados

# MÓDULO PYSPARK – AULA 1

* Apache Spark é uma ferramenta de processamento de dados (somente isso)
  + Tecnologia desenvolvida em Scala
  + Para rodar, necessário ter o JVM instalado
  + Ainda assim, não é necessário trabalhar com ele por Scala ou Java – também é possível trabalhar com Spark com SQL, Python (pela biblioteca PySpark) e R
    - Os códigos, nas diferentes linguagens, serão convertidos para um formato próprio, chamado RDD – depois serão processados em cluster pela Spark Engine



* O Apache Spark foi desenvolvido para processar dados massivos em cluster
  + Obs: o Spark não possui uma ferramenta de gerenciamento de cluster – necessário utilizar outras ferramentas com o Yarn (gerenciador de cluster do Radoop), Mesos e Kubernets
* Possível trabalhar tanto com SQL quanto com dataframes
* Os dados idealmente serão provenientes de um data lake, uma vez que o Spark não possui uma ferramenta de armazenamento

## PYSPARK NA PRÁTICA

* A máquina do Google Colab, por padrão, possui o Apache Spark já instalado, necessitando somente instalar a biblioteca PySpark, para poder manuseá-lo com Python
* Instalação biblioteca PySpark



* Criar sessão para executar comandos do PySpark

Texto

Descrição gerada automaticamente

* + Observação: somos nós que damos o nome do app (appName)
* Importando funções do PySpark



**Dataframes no Spark**

* Definir schema (colunas, passadas por uma lista) e dados (passados por tuplas dentro de uma lista)
* Depois é necessário definir um dataframe, passando como parâmetros os dados e o schema
* Obs.: os dataframes são imutáveis
  + Para modificá-lo, é necessário criar um novo dataframe ou sobrescrevê-lo

**Texto

Descrição gerada automaticamente**

* Verificar schema e tipos associados

**Texto

Descrição gerada automaticamente**

* + Obs: o Spark infere os tipos dos schemas, caso eles não sejam delimitados
* Determinar tipos das colunas

**Texto

Descrição gerada automaticamente**

* + Vantagem de determinar os tipos do schema: impede erros e consome menos processamento (Spark não precisa ler coluna inteira para inferir o tipo)
  + Obs.: Spark não é tão flexível – inserir o dado de um tipo errado gerará um problema
* Mostrar base de dados

**Texto

Descrição gerada automaticamente**

* + Obs.: o parâmetro passado dentro dos parênteses determina o número de linhas que serão apresentadas
* Retornar número de linhas no dataframe

**Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente**

* Retornar uma lista com as colunas do dataframe

**Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente**

**Transformações nos dados**

* 1ª transformação – SELECT

**Texto

Descrição gerada automaticamente**

* + Como parâmetros, devem ser passadas strings com o nome das colunas
  + Método .show() é obrigatório
  + Outra forma de selecionar colunas específicas (utilizando funções do PySpark)

**Texto

Descrição gerada automaticamente com confiança média**

* + Criando novo dataframe em cima dessa querry:

**Texto

Descrição gerada automaticamente**

* Aplicação de filtros no dataframe

**Texto

Descrição gerada automaticamente**

* + Repare que é esse método utiliza as funções do PySpark
  + Outro método:

**Texto

Descrição gerada automaticamente com confiança média**

* + Outro método:

**Texto

Descrição gerada automaticamente**

* Filtros compostos

Texto

Descrição gerada automaticamente

* + No PySpark, “ou” é simbolizado por | (barra reta)
  + “e” é simbolizado por &
* Utilizar SELECT e FILTER

**Texto

Descrição gerada automaticamente**

* + Nesse caso, apresenta-se nome, CPF e município de pessoas que vivem em fortaleza

# MÓDULO PYSPARK – AULA 2

**Importando dataframes a partir de arquivo CSV**

**Texto

Descrição gerada automaticamente**

**Visualizando dataframe sem truncar linhas**

* Se a informação for muito grande, o PySpark não a apresentará inteira, mas sim uma parcela seguida de três pontinhos
* Para que isso não aconteça, basta utilizar o parâmetro truncate = False na hora de chamar o método

****

## FILTROS

**Outra forma de se aplicar filtros**

**Texto

Descrição gerada automaticamente**

**Filtrar por substring**

**Texto

Descrição gerada automaticamente**

* Outro método de buscar a partir de uma substring (início da string)

**Texto

Descrição gerada automaticamente**

* Outro método de buscar a partir de uma substring (fim da string)

**Texto

Descrição gerada automaticamente com confiança média**

* Outro método para buscar a partir de uma substring (contém substring)

**Texto

Descrição gerada automaticamente**

## TRANSFORMAÇÃO DE COLUNAS

**Renomear colunas**

**Texto

Descrição gerada automaticamente**

* Necessário criar novo dataframe (lembrando: dataframes são imutáveis no Spark)
* Repare que primeiro é passado o nome da coluna original e depois o da coluna renomeada

**Criação de colunas**

**Interface gráfica do usuário

Descrição gerada automaticamente**

* Observação: esse comando não criou novo dataframe, logo ele não é definitivo
* Método: .withColumn()
* Primeiro parâmetro é o nome da nova coluna
* Segundo parâmetro é função que determina os valores da coluna
  + F.lit() permite passar um valor literal, uma string (todas as linhas possuirão mesmo valor literal nessa coluna)

## VISUALIZAÇÃO DE DADOS

**Dados numéricos sobre o dataframe**

**Texto

Descrição gerada automaticamente**

**Arredondar números**

* Utilizar F.round(F.col(‘coluna’), *número\_casas*)

**Mostrar registros únicos em uma tabela (distinct)**

**Texto

Descrição gerada automaticamente**

**Ordenar apresentação dos registros (order by / sort – crescente e decrescente)**

* O parâmetro passado é o nome da coluna que servirá para ordenação
* Repare que foram passadas duas colunas
  + Se o nome for igual, a segunda coluna servirá de parâmetro

**Texto

Descrição gerada automaticamente**

* Para determinar se será ordenado crescente ou decrescentemente, passa-se o método .asc() ou .desc() **na coluna** que estará ordenando a seleção

Texto

Descrição gerada automaticamente

* Pode-se utilizar a ordenação com o método .sort(), que terá o mesmo resultado

****

**Agrupar resultados (Group By)**

* Obs: sempre será adicionado um cálculo sobre o resultado agrupado
* Mostrar quantidade de resultados

**Texto

Descrição gerada automaticamente**

**Texto

Descrição gerada automaticamente**

* Filtrar por resultado e somar outra coluna

**Texto

Descrição gerada automaticamente**

**Texto

Descrição gerada automaticamente**

**Função de agregação para utilizar GroupBy (agg)**

* Permite usar diversas funções agregadas pelo mesmo GroupBy

Texto

Descrição gerada automaticamente

* + Repare que todos os cálculos estão relacionados ao campo “sexo”
  + Arredondando os valores “Total por sexo”:

Texto

Descrição gerada automaticamente

**Condições para atribuição de valores (when)**

* F.when(*condição***,** *valor*)
* .otherwise para passar uma condição contrária

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

* Para adicionar uma nova condição, antes do “.otherwise()”, basta adicionar novo “.when()”

## TRANSFORMAÇÃO DOS DADOS

* Separar string em substrings

Texto

Descrição gerada automaticamente

* + Primeiro parâmetro é a coluna que será percorrida
  + Segundo parâmetro é a posição (nesse caso, começa-se a contar a partir do 1 e não do 0)
  + Terceiro parâmetro é o tamanho da substring
  + Mesmos comandos, de forma suscinta:

Texto

Descrição gerada automaticamente

**Alterar valores dentro de uma coluna (regexp\_replace)**

* Observação: funciona somente com strings

Texto

Descrição gerada automaticamente

* Repare que não será criada uma nova coluna se o nome do primeiro atributo for o nome de uma coluna que já existe (no caso, a que será modificada)
* É possível modificar uma substring utilizando esse método
  + Exemplo: pretende-se retirar a substring “kg” dos valores da seguinte coluna

**Uma imagem contendo Texto

Descrição gerada automaticamente**

* + Para isso, basta passar como segundo parâmetro do método “regexp\_replace” a substring (isso porque esse método substitui o que for passado literalmente, não sendo necessário apresentar o valor inteiro – o que permite passar substrings)

**Interface gráfica do usuário, Texto

Descrição gerada automaticamente**

* Modificando com mais de uma condição ao mesmo tempo

**Texto

Descrição gerada automaticamente**

**Remover linhas duplicadas (dropDuplicates)**

**Interface gráfica do usuário

Descrição gerada automaticamente com confiança baixa**

**Remover colunas no dataframe**

**Texto

Descrição gerada automaticamente**

**Substituição de valores nulos**

**Interface gráfica do usuário, Texto

Descrição gerada automaticamente**

* É passado o valor que substituirá o campo nulo
* É possível definir subsets, ou seja, as colunas em que a substituição ocorrerá

**Uma imagem contendo Texto

Descrição gerada automaticamente**

# MÓDULO PYSPARK – AULA 3

## VISUALIZAÇÃO DE DADOS

* Visualização de tipos de dados

Texto

Descrição gerada automaticamente

## MANIPULAR DATAS (DATE)

* Converter uma data do formato string para o formato **date**

Texto

Descrição gerada automaticamente

* + Repare que basta sinalizar a coluna que contém as datas e passar o padrão em que foram inseridas
    - Obs: deve-se passar o mês como “MM” e não “mm” (pois “mm”, em caixa-baixa, corresponde a minutos)
  + A nova coluna terá datas no padrão YYYY-MM-DD
  + Mesma operação, mas sem criar nova coluna, somente substituindo originais:



* Converter padrão/formatação das datas

Texto

Descrição gerada automaticamente

* + Nesse caso, foi criada uma nova coluna
    - Caso desejasse corrigir a coluna original (lembrando que ela já deveria apresentar o tipo date), bastava apontar o nome da coluna antiga ao criar uma nova
  + Parâmetro: “F.date\_format(F.col(‘nome\_*coluna\_tipo\_date*’), 'dd/MM/yyyy')”
  + Observação importante: a nova coluna possuirá tipo string!
* Retornar a diferença em dias entre duas datas

Texto

Descrição gerada automaticamente

* + Necessário criar nova coluna
  + Passa parâmetro “F.datediff(data\_final, data\_inicial)”
* Retornar a diferença em meses entre duas datas

Texto

Descrição gerada automaticamente

* + Passa parâmetro “F.months\_between(data\_final, data\_inicial)”
* Retornar outros dados relacionados a datas
  + Utilizar seguintes métodos:



## SALVAR DATAFRAME (WRITE – SAVE)

**Comando básico**

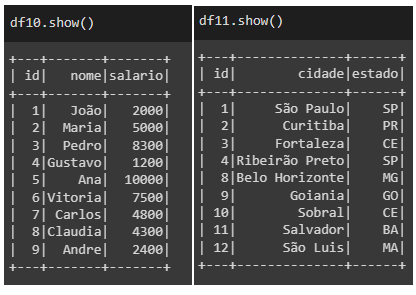


* Indicar formato e caminho (onde será salvo)
  + Formato padrão é ‘parquet’
  + Padrões aceitos: csv, parquet, json, orc
* PySpark aplica um nome para o arquivo
* Por padrão, o PySpark não sobrescreverá um dataframe que já foi salvo (que já existe na pasta destino), para mudar isso, deve-se passar o método .mode”(‘overwrite’)”
* Por padrão, o PySpark não salvará o nome das colunas (schema), para isso é necessário passar a opção (“header”, true)

Texto

Descrição gerada automaticamente

## CONEXÃO DE COLUNAS (JOINS)



* Repare que uma coluna possui mais registros que a outra. O **inner join** unirá somente colunas que possuem correspondências

Texto

Descrição gerada automaticamente

* Primeiro parâmetro aponta coluna que será apensada
* Parâmetro “on” determina coluna que fará a conexão
  + Caso o nome das colunas não fosse igual, deveria ser apontada a correspondência no comando:



* Parâmetro “how” determina tipo de join

**Left join**

* Considera todo conteúdo da tabela da esquerda importantes, devendo ser obrigatoriamente mostrados
* Left join faz com que registros que não possuem correspondência na tabela da direita apareçam com valores nulos

**Texto

Descrição gerada automaticamente**

**Right Join**

* Considera todo conteúdo da tabela da direita importantes, devendo ser obrigatoriamente mostrados
* Right join faz com que registros que não possuem correspondência na tabela da esquerda apareçam com valores nulos

**Texto

Descrição gerada automaticamente**

**Full Join/ Outer Join**

* Prioriza igualmente ambas as tabelas

**Texto

Descrição gerada automaticamente**

# MÓDULO SPARK – AULA 4

## CONEXÃO COM GOOGLE CLOUD STORAGE (DATA LAKE)

* Google Cloud Storage File System (GCSFS)
  + Biblioteca que permite conectar o Google Cloud Storage ao script Python

****

* + No COLAB, é necessário conectar-se ao drive por linha de código (para manusear arquivos)

**Texto

Descrição gerada automaticamente**

* + Importar acesso ao Google Cloud Storage

**Texto

Descrição gerada automaticamente**

* + Configurar chave de segurança que permite acesso ao Google Cloud Storage
    - Chave gerada no IAM/ Service Accounts (permitindo acesso ao Storage)
    - Chave armazenada no Colab

**Uma imagem contendo Texto

Descrição gerada automaticamente**

* + Acessar a Bucket e o arquivo armazenado nela

Texto

Descrição gerada automaticamente

* + - Necessário criar um objeto cliente
    - Caminho do arquivo (URI) é consultado dentro dos detalhes do arquivo na GCP

**Ler o dataframe**

**Texto

Descrição gerada automaticamente**

* Repare que é necessário passar uma configuração de conexão
  + “.config("spark.jars", 'https://storage.googleapis.com/hadoop-lib/gcs/gcs-connector-hadoop2-latest.jar')”
* Ler arquivo vindo do bucket

Texto

Descrição gerada automaticamente

## CONVERTER DATAFRAMES ENTRE SPARK E PANDAS

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

# MÓDULO SPARK – AULA 5

## TIPOS BÁSICOS DE DADOS DO SPARK

**Tipos básicos**

* ByteType -- Int
* ShortType -- Int
* IntegerType -- Int
* LongType -- Int
* FloatType -- Float
* DoubleType -- Float
* StringType -- String
* BooleanType -- Bool
* DecimalType -- Float
* NULL

Para mais informações: <https://spark.apache.org/docs/latest/sql-ref-datatypes.html>

**Tipos complexos**

* TimestampType – DateTime
* DateType – DateTime
* ArrayType – Lista, tupla, array
* MapType – Dict
* StructType – Lista, tupla
* StructField – Tipo de campo

## STRUCT TYPES

* Definir os tipos das colunas, ao invés de solicitar ao Spark que infira o schema
  + Os dados inseridos que não forem do tipo determinado entrarão como nulos (o que pode ser interessante para se procurar inconsistências nas linhas – funcionando como uma espécie de validação)

**Configuração**

* Necessário importar os tipos de variáveis

Texto

Descrição gerada automaticamente

* + O asterisco significa que todos os tipos serão importados
  + É possível delimitar, no lugar do asterisco, quais tipos serão importados
* Montando o schema

Texto branco sobre fundo preto

Descrição gerada automaticamente

* + Dentro do StructType, necessário definir um StructType por coluna
    - Primeiro parâmetro é o nome do campo
      * É possível renomear já nesse momento o nome da coluna
    - Segundo, tipo da variável
    - Terceiro, se pode ser nulo
* Ler o dataframe

Texto

Descrição gerada automaticamente

* + Repare que a opção inferschema foi marcada como falsa
  + Repare que, ao carregar o dataframe, apontou-se o esquema das colunas

## WINDOW FUNCTIONS

* Criam janelas de verificação
* Necessário importar módulo Window



* Necessário criar uma partição

Texto

Descrição gerada automaticamente

* + Separa por grupo e depois analisa todas as linhas, retornando a linha que corresponde ao critério passado
  + Parâmetro do método ‘.partitionBy()’ é a coluna que servirá para classificação
  + Chamado método ‘.orderBy()’ para criar a classificação (nesse caso, se está selecionando maior salário por departamento)
* Row Number (apresenta o número das linhas)

Texto

Descrição gerada automaticamente

* + Repare que foi criada uma coluna que ordena do menor para o maior salário por departamento
* Rank (apresenta um ranking)

Texto

Descrição gerada automaticamente

* + Para efeitos de visualização, nesse caso passou-se o orderBy como sendo decrescente

Texto

Descrição gerada automaticamente

* + Repare que ele apresentou os valores empatados, depois pulando para o sétimo
    - Se quiser que seja pulado para o terceiro (e não o sétimo), utiliza-se a função “F.dense\_rank()”

Texto

Descrição gerada automaticamente

* Lag (apresenta valores anteriores, de acordo com o número determinado de registros)

Texto

Descrição gerada automaticamente

* + Repare que é passada uma coluna e o número de registros anteriores, que serão apresentados
* Lag (apresenta próximos valores, de acordo com o número determinado de registros)

Texto

Descrição gerada automaticamente

## SPARK SQL

* Possível executar SQL pelo Spark
* Depois de montar o dataframe, deve-se utilizar os comandos SQL dentro de spark.sql(‘*códigos\_SQL*’).show()

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

**Criar tabela a partir do dataframe**

* Necessário salvar o dataframe como tabela

**Texto

Descrição gerada automaticamente**

**Texto

Descrição gerada automaticamente**

**Criar dataframe a partir de seleção (SELECT) no SQL**

**Texto

Descrição gerada automaticamente**

## CONECTANDO MYSQL AO PYTHON

* Necessário instalar a biblioteca MySQL Connector



Interface gráfica do usuário

Descrição gerada automaticamente

* Conectar-se com o banco de dados

Texto

Descrição gerada automaticamente

* + Servidor será ‘localhost’ se for na própria máquina; se estiver em um servidor, será o IP da máquina
* Utilizar comandos SQL
  + Leitura de dados de um banco de dados
    - Necessário criar uma função que permitirá passar os comandos

Texto, Aplicativo

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

* + Executar SQL (INSERT, DELETE, UPDATE)

Texto

Descrição gerada automaticamente

**Transformando SQL em dataframe no Pandas com Python**

**Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente**

# LOOKER STUDIO (ANTIGO DATA STUDIO)

* Visualizador de dados/ criador de dashboards
* Outras ferramentas que criam dashboards: Tableau/ PowerBi/ DataPrep/ Looker (Desktop)
* Recomendado fazer o tratamento dos dados antes de criar o dashboard

**Funções do Looker Studio**

**Texto

Descrição gerada automaticamente**

* Permite fazer dashboards com bases de dados diferentes (mesmo dashboard apresenta bases diferentes, permitindo traçar insights mais complexos)
* Possível designar usuários que serão somente visualizadores e outras que serão editores

**Texto

Descrição gerada automaticamente**

## LOOKER STUDIO NA PRÁTICA

* Iniciar selecionando “Fonte de dados”, para abrir a base de dados e depois fazer dashboard
* Escolher conector, seguir instruções e conectar
* Escolher tipos das colunas
  + Qualquer mudança proposta no Looker Studio só terá efeito na própria plataforma (não altera base de dados original)
* Criar relatório

# DATAPROC

* Produto da Google Cloud que inclui:
  + LoadBalance, VPC, Hadoop, Spark, Python, Jupyter Notebook e Google Storage
  + Trabalho em cluster
* Solução completa para criação, manutenção e consulta de bases de dados BigData
* Rápida configuração, alta volatilidade (custo bastante elevado)
  + Deve ser utilizado por poucas horas, o bastante para conseguir trabalhar na base de dados (depois, o trabalho será enviado para outras soluções, como Cloud Storage)

## CONFIGURANDO DATAPROC

* Definir políticas de autoescalonamento
  + Escala tanto a nível de máquina quanto software (ler características dos tipos de escalonamento)
    - Preferível utilizar “MapReduce” se não for utilizar streaming de dados
  + Definir nome da política
  + Definir região em que será montado o cluster (observação: nem todas as regiões possuirão todas as máquinas)
  + Configuração de YARN e fator de escalonamento: manter padrão
  + Worker Configuration:
    - Primary: determina de quantos em quantos nodes serão criados
    - Secondary: determina número mínimo e máximo de nodes
* Criar Cluster
  + Escolher criar por meio do Compute Engine ou de Kubernets
    - Por padrão, utilizar Compute Engine
  + Determinar nome (único no projeto)
  + Escolher localização (evitar us-central1)
  + Escolher tipo do cluster
    - Sempre criar, pela primeira vez, como Single Node (1 mestre e 0 nós)
      * Nesse cenário, o próprio mestre trabalha
      * Bom para trabalhar com até 10 milhões de linhas e 50 colunas
      * Não há escalonamento (somente trabalha com uma máquina, mas incorpora todas as demais soluções da ferramenta – ambiente configurado e pronto para uso)
    - Opção Standard só funciona se houver política de autoescalonamento
  + Escolher política de autoscaling que já foi determinada
  + Escolher a imagem do Dataproc (versão)
    - Preferir utilizar sistema baseado em Debian – seguro e consome pouca máquina
  + Habilitar componente gateaway (Enable component gateway)
    - Permite acessar o cluster por aplicação web (sempre habilitar)
  + Selecionar serviços incluídos no Dataproc
    - No mínimo, instalar o Jupyter e o Zappelin
  + Escolher máquina (da master e dos nodes – serão os mesmos)
    - Iniciar na n1-standard-2 (melhor custo-benefício para projetos pequenos)
    - Escolher tamanho do disco (padrão é 500GB)
      * Escolher se HDD, SSD ou híbrido
    - ‘Total YARN usage’ corresponde à capacidade de processamento de dados da máquina (já desconsidera o utilizado pelo sistema/master)
  + Configurações de rede
    - Necessário criar, anteriormente, uma VPC somente para o cluster
    - Marcar ‘Internal IP only’ se quiser restringir o acesso ao cluster somente às máquinas da Google Cloud
  + Configurar política de desligamento do cluster (Scheduled deletion)
    - Submeter tempo fixo para que seja deletado (i.e. 10 horas depois de criado)
    - Submeter “timeout”, geralmente de 1 hora (em outras palavras, será deletado depois de uma hora ocioso)
  + Na parte de segurança, habilitar Project access (permite acessar outros recursos do seu projeto no Google Cloud)
* Código para criação desse Dataproc no shell: gcloud dataproc clusters create cluster-8a4b --enable-component-gateway --region us-east1 --zone us-east1-b --single-node --master-machine-type n1-standard-2 --master-boot-disk-size 200 --image-version 2.0-debian10 --optional-components JUPYTER,ZEPPELIN --max-idle 3600s --max-age 1200s --scopes 'https://www.googleapis.com/auth/cloud-platform' --project bc26-marcos-valente

# PIPELINES

* Pipelines dizem respeito a fluxos de trabalho (extração, tratamento e devolução)
* Nem todas as pipelines são automatizadas
  + Não automatizado: utilizamos Pandas e PySpark
  + Automatizados também utilizam Pandas e PySpark, mas com processos agendados, pré-configurados
* Havendo repetição do processo, é possível automatizar a pipeline
  + No entanto, é necessário verificar, periodicamente, se a automatização do processo continua válida, uma vez que a estrutura das tabelas pode ser alteradas
* Ferramentas:
  + Apache Airflow (visual, necessário subir uma VM para funcionar na nuvem)
  + Apache Beam (código e visual, com melhor integração com as nuvens)
  + Apache Hoop

**Apache Beam**

* Cria modelos de automação unificados
* Funciona da mesma forma em qualquer nuvem
* Pode ser escrito em três linguagens: Java, Python, Go
* Pipelines com Python, Java, Scala, R, SQL...

## APACHE BEAM NA PRÁTICA

* Seguir esta ordem (no colab)!
* No colab, é necessário dar upgrade no Pip e reiniciar ambiente de execução

Texto

Descrição gerada automaticamente

* Instalar Apache Beam do tipo interativo e reiniciar ambiente de execução

****

* Instalar o Apache Beam para GCP e reiniciar ambiente de execução

****

* Importar o Apache Beam

**Texto

Descrição gerada automaticamente**

* Criar variável da pipeline

**Texto

Descrição gerada automaticamente com confiança média**

**Criação do modelo de pipeline**

**Texto

Descrição gerada automaticamente**

* + Observação: todo passo do pipeline deve ser indicado por uma barra em pé ( | ), a qual deve ser sempre descrita (string ao lado do passo)
    - Ao lado da descrição, deve ser indicado o comando que será executado
  + beam.io = “Apache Beam imput/output”
  + CSV é considerado um arquivo de texto para o Apache Beam
  + Método de Map mapeia o dataframe (o percorre)
    - Para cada registro, o dividirá sempre que encontrar uma vírgula
* Executar a pipeline

****

* A primeira linha de dados de um CSV é sempre o nome das colunas
  + Nesse caso, entrará da mesma forma que os outros dados, como se fosse uma lista (para o Apache Beam, cada linha do dataframe é uma lista)
    - Mostra os dados de forma literal
    - Busca são realizadas no índice da lista
* Para fazer cálculos com dados vindos do Apache Beam, é necessário pular a primeira linha (o nome das colunas, pois elas trarão dados em string)

**Texto

Descrição gerada automaticamente**

* + Se o atributo passado for ‘0’, ao invés de ‘1’, o nome das colunas serão inseridas

**Salvar arquivos**

**Texto

Descrição gerada automaticamente**

* É possível salvar como CSV
* Não salvar depois de imprimir os dados, pois as linhas virão vazias (print somente para verificar como o dataframe está vindo)

**Filtros**

****

Texto

Descrição gerada automaticamente

* Nesse dataframe, a posição da coluna que apresenta aeroportos é a “0” (primeira coluna)
* Filtro baseado em números

Logotipo

Descrição gerada automaticamente



* + Repare que o tipo do dado foi convertido para float antes de fazer a comparação

**Agrupamento**

* Agrupando e somando

****

**Texto

Descrição gerada automaticamente**

* + Repare que a chave utilizada será o aeroporto (campo [1] da tabela)
  + Resultado:

**Texto

Descrição gerada automaticamente**

* Agrupando e contando

****

**Texto

Descrição gerada automaticamente**

**Texto, Tabela

Descrição gerada automaticamente**

* Agrupar duas informações diferentes (mas que possuem chaves iguais)
  + Necessário criar duas pipelines, que rodarão em paralelo
  + Depois é necessário criar uma nova tabela, apresentando os dados no formato dicionário e juntando-os pela chave

**Texto

Descrição gerada automaticamente**

**Texto branco sobre fundo preto

Descrição gerada automaticamente**

## CONEXÃO APACHE BEAM – GOOGLE CLOUD

* Criar bucket com pastas ‘models’, ‘temp’, ‘staging’
* Definir configurações

Texto

Descrição gerada automaticamente

* Montar os pipelines

Texto

Descrição gerada automaticamente

* Essa programação **cria o modelo** que será rodado pelo DataFlow
  + Ele não cria o novo dataframe!

**Trabalhando com Dataflow**

* Dataflow cria clusters de processamento, executando ações baseadas em modelo (software as a servisse)
  + Custo financeiro está baseado somente no processo executado
* Entrar no Dataflow
* Create job from template
  + Indicar nome do processo (boa prática: indicar se é batch ou streaming)
  + Indicar o template
    - Custom Template
    - Passar caminho do template no bucket
    - Passar uma localização para arquivos temporários (a mesma setada no setup do template)
  + Run Job
    - Dataflow irá iniciar o trabalho e alocar o número de máquinas necessárias
    - Quando o trabalho for concluído, o cluster será desmontado

# BIG QUERRY

* Trata-se do Data Warehouse provido pela Google Cloud
* O Data Warehouse permite criar tabelas temporárias (a partir de uma base de dados original), as quais poderão ser consumidas pelos interessados e desmontadas (otimizando o custo)
  + Essas tabelas (ou conjunto de tabelas) são chamadas Datasets
  + Não se altera a base de dados original
  + Permite extração de insights por analistas (parte analítica)
* Destinado a receber dados tratados e padronizados

## CRIANDO DATASET

- Criando Dataset

- BigQuery

- Três pontinhos do projeto

- Create Dataset

- DatasetID (único no projeto e relacionado aos dados)

- Escolher data location (a mesma do seu bucket)

- Setar data expiration para não esquecer de excluí-las depois

- Advanced options (default)

- Create dataset

- Criando Table

- Três pontinhos do dataset

- Create table

- Source - create table from: GCS - selecionar arquivo a ser usado no bucket - file format: conforme seleciona o arquivo ele muda automaticamente

- Destination - project: seu projeto - dataset: o dataset criado - nomear table - table type: native

- Schema: Auto detect

- Partition and cluster settings: particiona dados para melhorar tempo e preço (mas não iremos particionar)

- Advanced options - Write preference: default - Number of errors allowed: default - Field delimiter: nesse caso foi customizado (:) - habilitar quoted newlines - Header rows to skip: default (0) - Encryption: Google-managed - Default Collation: default

- Create table (verificar sempre se a dataset está na mesma localização de seu bucket, se não da erro)