**Projeto: Modelagem de Dados com Postgres**

**Introdução**

Uma startup chamada Sparkify deseja analisar os dados que eles coletaram sobre as músicas e a atividade do usuário em seu novo aplicativo de streaming de música. A equipe de análise está particularmente interessada em entender quais músicas os usuários estão ouvindo. Atualmente, eles não têm uma maneira fácil de consultar seus dados, que residem em um diretório de logs JSON na atividade do usuário no aplicativo, bem como em um diretório com metadados JSON nas músicas em seu aplicativo.

Eles gostariam que um engenheiro de dados criasse um banco de dados Postgres com tabelas projetadas para otimizar as consultas sobre a análise da reprodução da música e trazê-lo para o projeto. Sua função é criar um esquema de banco de dados e um pipeline de ETL para essa análise. Você poderá testar seu banco de dados e pipeline de ETL executando consultas fornecidas a você pela equipe de análise do Sparkify e comparar seus resultados com os resultados esperados.

**Descrição do Projeto**

Neste projeto, você aplicará o que aprendeu na modelagem de dados com Postgres e construirá um pipeline de ETL usando Python. Para concluir o projeto, você precisará definir tabelas de fatos e dimensões para um esquema em estrela para um foco analítico específico e escrever um pipeline ETL que transfere dados de arquivos em dois diretórios locais para essas tabelas em Postgres usando Python e SQL.

# Conjunto de dados de música

O primeiro conjunto de dados é um subconjunto de dados reais do [Conjunto](https://labrosa.ee.columbia.edu/millionsong/) de dados do [milhão de canções](https://labrosa.ee.columbia.edu/millionsong/) . Cada arquivo está no formato JSON e contém metadados sobre uma música e o artista dessa música. Os arquivos são particionados pelas três primeiras letras do ID da faixa de cada música. Por exemplo, aqui estão os caminhos de arquivo para dois arquivos neste conjunto de dados.

song\_data/A/B/C/TRABCEI128F424C983.json

song\_data/A/A/B/TRAABJL12903CDCF1A.json

E abaixo está um exemplo da aparência de um único arquivo de música, TRAABJL12903CDCF1A.json.

{"num\_songs": 1, "artist\_id": "ARJIE2Y1187B994AB7", "artist\_latitude": null, "artist\_longitude": null, "artist\_location": "", "artist\_name": "Line Renaud", "song\_id": "SOUPIRU12A6D4FA1E1", "title": "Der Kleine Dompfaff", "duration": 152.92036, "year": 0}

**Conjunto de dados de registro**

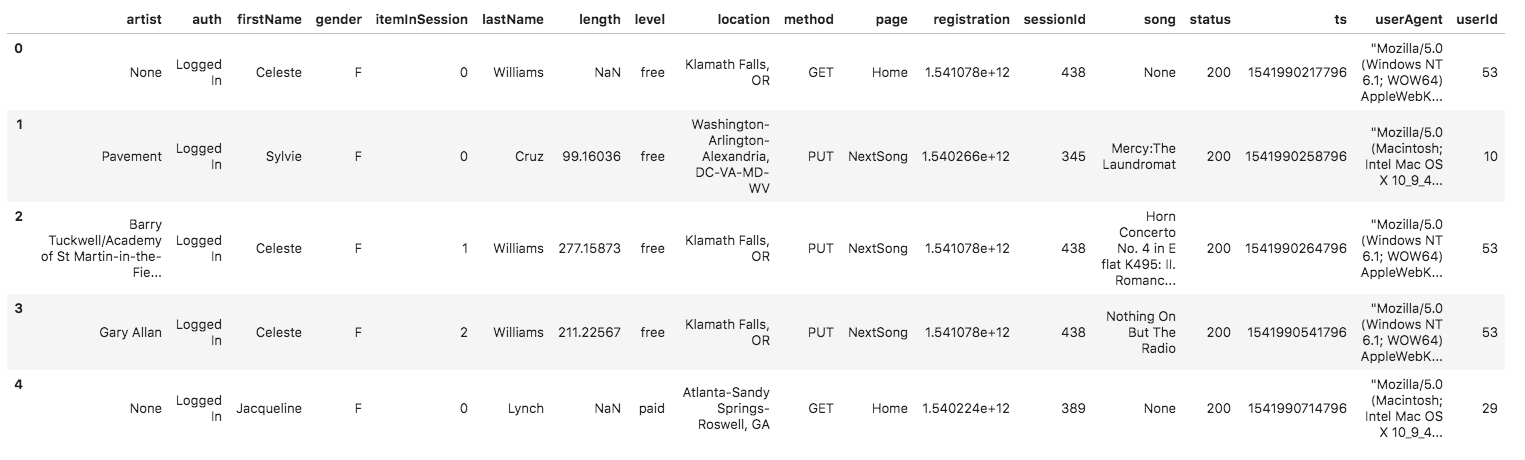
O segundo conjunto de dados consiste em arquivos de log no formato JSON gerados por este [simulador de eventos com](https://github.com/Interana/eventsim) base nas músicas do conjunto de dados acima. Eles simulam registros de atividades de um aplicativo de streaming de música com base em configurações especificadas.

Os arquivos de log no conjunto de dados com o qual você trabalhará são particionados por ano e mês. Por exemplo, aqui estão os caminhos de arquivo para dois arquivos neste conjunto de dados.

log\_data/2018/11/2018-11-12-events.json

log\_data/2018/11/2018-11-13-events.json

E abaixo está um exemplo de como são os dados em um arquivo de log, 2018-11-12-events.json.



Se quiser ver os dados JSON nos arquivos log\_data, você precisará criar um dataframe do pandas para ler os dados. Lembre-se de primeiro importar as bibliotecas JSON e pandas.

df = pd.read\_json(filepath, lines=True)

Por exemplo,

df = pd.read\_json('data/log\_data/2018/11/2018-11-01-events.json', lines=True)

leria o arquivo de dados 2018-11-01-events.json.

Caso você precise se atualizar sobre os formatos de arquivo JSON, [aqui está um vídeo útil](https://www.youtube.com/watch?time_continue=1&v=hO2CayzZBoA) .

# Esquema para Análise de Reprodução de Música

Usando os conjuntos de dados de música e registro, você precisará criar um esquema em estrela otimizado para consultas na análise de reprodução de música. Isso inclui as tabelas a seguir.

#### Tabela de Fatos

1. **Songplays** - registros em dados de log associados com músicas **tocadas,** ou seja, registros com páginaNextSong
   * songplay\_id, start\_time, user\_id, nível, song\_id, artist\_id, session\_id, localização, user\_agent

#### Tabelas Dimensionais

1. **usuários** - usuários no aplicativo
   * user\_id, first\_name, last\_name, gênero, nível
2. **canções** - canções no banco de dados de música
   * song\_id, title, artist\_id, ano, duração
3. **artistas** - artistas em banco de dados de música
   * artist\_id, nome, localização, latitude, longitude
4. **tempo** - carimbos de data / hora de registros em **canções** divididos em unidades específicas
   * start\_time, hour, day, week, month, year, weekday

# Modelo de Projeto

Para iniciar o projeto, vá para a área de trabalho na próxima página, onde você encontrará os arquivos de modelo de projeto. Você pode trabalhar em seu projeto e enviar seu trabalho por meio deste espaço de trabalho. Como alternativa, você pode baixar os arquivos de modelo de projeto da pasta Recursos se desejar desenvolver seu projeto localmente.

Além dos arquivos de dados, o espaço de trabalho do projeto inclui seis arquivos:

1. test.ipynb exibe as primeiras linhas de cada tabela para permitir que você verifique seu banco de dados.
2. create\_tables.pydescarta e cria suas tabelas. Você executa esse arquivo para redefinir suas tabelas antes de cada vez que executa seus scripts ETL.
3. etl.ipynblê e processa um único arquivo de song\_datae log\_datae carrega os dados em suas tabelas. Este bloco de notas contém instruções detalhadas sobre o processo ETL para cada uma das tabelas.
4. etl.pylê e processa arquivos de song\_datae log\_datae os carrega em suas tabelas. Você pode preencher isso com base em seu trabalho no bloco de notas ETL.
5. sql\_queries.py contém todas as suas consultas sql e é importado para os três últimos arquivos acima.
6. README.md fornece discussão sobre seu projeto.

# Etapas do Projeto

Abaixo estão as etapas que você pode seguir para concluir o projeto:

### Criar tabelas

1. Escreva CREATEinstruções sql\_queries.pypara criar cada tabela.
2. Escreva DROPinstruções sql\_queries.pypara eliminar cada tabela, se houver.
3. Execute create\_tables.pypara criar seu banco de dados e tabelas.
4. Execute test.ipynbpara confirmar a criação de suas tabelas com as colunas corretas. Certifique-se de clicar em "Reiniciar kernel" para fechar a conexão com o banco de dados após executar este notebook.

### Construir processos ETL

Siga as instruções do etl.ipynbbloco de notas para desenvolver processos ETL para cada tabela. No final de cada seção da tabela, ou no final do bloco de notas, execute test.ipynbpara confirmar se os registros foram inseridos com sucesso em cada tabela. Lembre-se de executar novamente create\_tables.pypara redefinir suas tabelas antes de cada vez que você executar este notebook.

### Construir pipeline de ETL

Use o que você completou etl.ipynbpara completar etl.py, onde você processará todos os conjuntos de dados. Lembre-se de correr create\_tables.pyantes etl.pyde reiniciar suas tabelas. Execute test.ipynbpara confirmar se seus registros foram inseridos com êxito em cada tabela.

### Processo de Documento

Execute as seguintes etapas em seu README.mdarquivo.

1. Discuta o propósito deste banco de dados no contexto da inicialização, Sparkify e seus objetivos analíticos.
2. Declare e justifique o design do esquema do banco de dados e o pipeline de ETL.
3. [Opcional] Forneça consultas e resultados de exemplo para análise de reprodução de música.

Aqui está um [guia](https://www.markdownguide.org/basic-syntax/) sobre a sintaxe de Markdown.

### ****NOTA:**** Você não poderá executar test.ipynb, etl.ipynbou etl.pyaté que tenha executado create\_tables.pypelo menos uma vez para criar o sparkifydbbanco de dados ao qual esses outros arquivos se conectam.

# Rubrica do Projeto

Leia a [avaliação](https://review.udacity.com/#!/rubrics/2500/view) do projeto antes e durante o desenvolvimento de seu projeto para garantir que você atenda a todas as especificações.