Parte 1 - SQL

Dado o schema de dados fornecido, escreva queries SQL executáveis para responder às questões abaixo. Por favor elabore suas respostas em uma única query, e assuma acesso somente leitura ao banco de dados (ex.: não use CREATE TABLE).

Para os últimos 30 dias, deduza a diferença média e mediana entre Actual e Predicted ETA de todas as viagens feitas na cidade de ‘Sao Paulo’ e ‘Campinas’. (ETA: Estimated Time of Arrival, ou Tempo Estimado até Chegada)

Um evento é criado na tabela events com um timestamp sempre que um passageiro tenta se cadastrar (com um event_name 'attempted_sign_up') ou se cadastra com sucesso (com um event_name 'sign_up_success'). Para todos os passageiros que se cadastraram com sucesso nas cidades 'Sao Paulo' e 'Campinas' na primeira semana de 2018, encontre em cada cidade, para cada dia da semana (da data de cadastro), a porcentagem de passageiros que completam uma viagem em até 168 horas após a hora de cadastro.

Assuma um banco PostgreSQL, timezone do server é UTC.

Tabela: **Trips**

Coluna:	Datatype:
id	integer
client_id	integer (Foreign keyed to events.rider_id)
driver_id	integer
city_id	integer(Foreign keyed to cities.city_id)
client_rating	integer
driver_rating	integer
request_at	Timestamp with timezone
predicted_eta	Integer
actual_eta	Integer

status	Enum(‘completed’, ‘cancelled_by_driver’, ‘cancelled_by_client’)

Tabela: **cities**

Coluna:	Datatype:
city_id	integer
city_name	string

Tabela: **events**

Coluna:	Datatype:
device_id	integer
rider_id	integer
city_id	integer
event_name	Enum(‘sign_up_success’, ‘attempted_sign_up’,‘sign_up_failure’)
_ts	Timestamp with timezone

Parte 2 Transformação de dados

O time de Data da Caixa Seguradora precisa de um repositório de dados acessível para a elaboração de relatórios através de queries SQL. Para isso, lhe é pedido que você crie uma

rotina em batch simples que seja capaz de processar dados não-estruturados, hoje em forma de eventos JSON, e os transforme em

uma estrutura relacional de banco de dados usando SQLite. Os arquivos JSON foram fornecidos junto deste exercício, dentro do arquivo routing-keys.zip.

Você sabe *a priori* que os dados são provenientes de dois principais microsserviços: order-service e policy-service, microsserviços responsáveis por cotações e apólices de seguro, respectivamente. Cada microsserviço emite eventos em diferentes filas de mensagem, aqui denominadas routing keys. Ou seja, cada fila representa um evento (ou acontecimento) diferente, seja esse a emissão de uma apólice ou criação de uma cotação, dentre outros.

Cada routing key contém em seu nome:

- O nome do microsserviço que emitiu aquele evento
- O produto a qual aquele evento se refere (auto, home ou life, para seguros de automóvel, residência ou vida, respectivamente)
- A ação à qual aquele evento se refere - por exemplo, policy.created representa a criação de uma apólice.

O time de DevOps da Caixa Seguradora disponibilizou acesso a uma pasta contendo os dados não-estruturados de cada um dos eventos emitidos, em formato JSON, organizados e separados em diferentes arquivos por routing key, onde cada linha do arquivo representa um evento emitido. Você deve:

1. Analisar a estrutura dos eventos de cada routing key e possíveis relacionamentos entre elas, para que possa compreender o fluxo de dados e elaborar o modelo relacional.

2. Construir, usando o spark, um programa que seja capaz de receber como input todos os arquivos .json fornecidos, e tenha como output um banco de dados relacional SQLite organizado em diferentes tabelas, contendo os dados fornecidos, porém em formato estruturado, de forma que analistas da Caixa Seguradora possam rodar queries SQL e analisar os dados. Toda a criação de tabelas deve ser feito pelo seu próprio programa, e não manualmente. Amamos testes em nossos projetos

3. Nos envie de volta o seu código completo, assim como o arquivo .sqlite gerado pelo seu programa após a execução.

Descrição de campos nos eventos:

routing_key*: routing key, ou nome de fila, à qual aquele evento pertence. A routing key é composta de: nome do microsserviço que emitiu o evento, produto a qual o evento se refere (auto, home ou life), e ação à qual o evento se refere (por exemplo, policy.created)*

message_id*: ID único do evento. Esse ID possui garantia de unicidade, e mesmo que um evento seja emitido repetidamente, o seu message_id não vai ser duplicado.*

raw_timestamp*: Timestamp em formato unixtime, representando o momento em que o evento foi emitido*

order_uuid*: ID único da cotação*

insurance_type*: Produto (Auto, Home ou Life)*

sales_channel*: Canal de venda por onde a cotação foi iniciada*

lead_person*: dados pessoais do cliente que está cotando o seguro*

monthly_cost*: valor a ser pago pelo cliente mensalmente sobre seu seguro*

reason*: motivo ou descrição do evento ocorrido*

payload*: conteúdo do evento emitido*

Todos os dados fornecidos são falsos e não representam comportamento real de clientes Caixa Seguradora.

Parte 3 - **Arquitetura**

Queremos que você fale um pouco sobre a organização da arquitetura necessária para atender uma necessidade de negócio. Vamos falar de um caso de uso real e gostaríamos que propusesse uma solução envolvendo as ferramentas com que tem familiaridade. Sinta-se a vontade para explicar de maneira descritiva, com desenhos ou qualquer forma que preferir, o importante mesmo é explicar quais são suas ideias.

A plataforma de vendas da Youse foi criada usando o paradigma de microsserviços, existem vários sistemas responsáveis por partes diferentes da jornada de vendas e interações feitas com o usuário durante o seu tempo como cliente. Cada serviço tem uma responsabilidade diferente e um banco de dados exclusivo para si, dessa maneira, os dados da Youse ficam separados em vários serviços com modelagens diferentes.
Embora a persistência dos dados seja feita de maneira distribuída, também fazemos uso do RabbitMQ como ferramenta para comunicação assíncrona entre os serviços usando os paradigmas de mensageria e event-sourcing, isso nos permite ter acesso em tempo real dos eventos processados pela plataforma.

Precisamos de um pipeline de dados que permita o relacionamento dos diferentes dados da empresa para criação de dashboards e análise diariamente.