Desafio Engenheiro de Dados

Autor: Anselmo Oliveira

Data: 14/11/2024

Desafio: Desenvolver Arquitetura de Dados

Introdução

O foco principal deste desafio, tanto na parte teórica quanto prática, foi a utilização de soluções escaláveis, adotando processos que podem ser ampliados em ferramentas como bancos de dados, Python e SQL. O projeto foi desenvolvido com base no conceito de ELT (Extract, Load, Transform), ao invés do tradicional ETL (Extract, Transform, Load), visando uma ingestão de dados que favorece a modelagem por camadas, com menor custo computacional e maior eficiência em grandes volumes de dados. Essa abordagem também reduz a complexidade do processo.

A escolha das ferramentas, como Databricks (para o Data Lake) e Kafka (como ferramenta de ingestão), possibilitou a execução dos exercícios práticos em Python e SQL, utilizando o Spark no Databricks, o que otimiza o desempenho e garante a escalabilidade da solução. A utilização de streams de dados com Spark e Databricks é altamente vantajosa, permitindo processar volumes massivos de dados de maneira eficiente.

O Kafka, por sua vez, é capaz de escalar verticalmente, permitindo a ingestão de milhões de registros conforme necessário. Além disso, o versionamento de dados no Databricks e o uso de cache proporcionam maior controle e desempenho nas operações, garantindo flexibilidade e agilidade no processo de análise de dados.

Ferramenta de ingestão:

Kafka: Foi selecionado devido ao seu desempenho com dados em stream, documentação, performance e escalabilidade.

Airbyte: Trabalha bem com dados em stream e oferece diversos componentes para ingestão de dados, mas o Kafka é mais estável, performático e escalável para cenários de grandes volumes de dados.

Airflow: Produto não foi nativamente projetado para trabalhar com streaming de dados.

Dagster: Produto não foi nativamente projetado para trabalhar com streaming de dados. Além disso, por ser uma ferramenta open-source e com atualizações frequentes, pode exigir um esforço considerável para mantê-la.

Ferramenta de Datalake

Databricks: Foi selecionado devido a maturidade da ferramenta, documentação e performance. Os recursos de stream, custos, cache e integrações com recursos como DVC, Noteboks, MLflow e entre outros.

Snowflake: É um ótimo plataforma de dados diversos recursos como: integração com diversas ferramentas, escala, Noteboks, cache, governança entre outros. No entanto O databricks é mais completo em relação a integração com produtos de cloud como Azure e AWS.

Iceberg: Plataforma open-source que possui a maioria dos recursos do deltalake do databrikcs. No entanto, a responsabilidade pela atualização da plataforma, integração de recursos e gerenciamento de versões das dependências pode gerar um custo alto de sustentação, o que pode ser um desafio á longo prazo.

Ferramenta de Dataviz:

Power BI: O consumo de dados diretamente de um Data Lake pode gerar custos elevados de processamento e deve ser evitado sempre que possível. É comum utilizar um banco de dados intermediário para essa finalidade, aplicando técnicas como cargas incrementais ou streams. A escolha do Power BI se justifica pelo recursos disponíveis, como mapas, gráficos e a responsividade mobile. Sua arquitetura em nuvem, que permite armazenar dados importados de fontes remotas, garante segurança e desempenho.

Metabase: Ferramenta open-source que oferece recursos como criação de gráficos, dashboards e governança de dados. No entanto, enfrenta limitações em dispositivos móveis, o que pode impactar a experiência do usuário.

Superset: Ferramenta open-source que também oferece recursos robustos, como criação de gráficos, dashboards e governança de dados. Assim como o Metabase, enfrenta desafios em dispositivos móveis, o que pode prejudicar a usabilidade.

Questão 1

Projete uma solução para uma plataforma de comunicação que computa a utilização de notificações Whatsapp de milhares de clientes a cada minuto.

Projete uma solução para uma plataforma de comunicação que computa a utilização de notificações Whatsapp de milhares de clientes a cada minuto.

A interface gráfica é um painel simples, apresentando alguns dados conforme o exemplo abaixo:

Painel de Notificações
`Cliente: Acme`
`Notificações Whatsapp: 5000`
`Última atualização: 2020-02-03T15:00:00+00:00`

As notificações são processadas a cada minuto e podem ser exportadas em um relatório detalhado conforme exemplo abaixo:

Relatório Detalhado de Notificações
`= Cliente: Acme =`
`Mensagem ID: 5B81FF24A1`
`Payload: “Olá XXXXX, seja muito bem-vindo! Precisa de alguma ajuda?”`
`Data hora: 2020-02-03T14:00:01.001+00:00`
...

`Mensagem ID: A61D178E73`
`Payload: “Não identificamos a sua última mensagem. Por favor, entre em contato”`
`Data hora: 2020-02-03T14:00:01.099+00:00`
...

Além disso, a solução também deve possibilitar a geração de um relatório de cobrança uma vez por dia, conforme exemplo abaixo:

Relatório de Cobrança
`Cliente: Acme`
`Notificações Whatsapp: 25000`
`Valor: R$ 1000,00`
`Data: 2020-02-03`

A origem dos dados é um tópico no Kafka, com um volume de 100 milhões de mensagens/dia, com o seguinte payload:

customer_id	message_id	customer_name	channel	payload	event_datetime
7494212	5B81FF24A1	Acme	Whatsapp	Olá XXXXX, seja muito bem-vindo! Precis...	2020-02-03T14:00:01.001+00:00
7494212	A61D178E73	Acme	Whatsapp	Não identificamos a sua última mensagem...	2020-02-03T14:00:01.099+00:00

Proponha uma arquitetura que contemple cada um dos casos de uso: painel de notificação, relatório detalhado de notificações e relatório de cobrança.

Resposta da Questão 1

Resumo:

A plataforma mencionada no enunciado descreve um sistema de PAS (Plataforma como Serviço), que fornece para os clientes de marketing como o envio de nansagens para seus leads. O painel é muito importante para o cliente conseguir fazer relatórios, valizações, investigações e calculos de custos. A Origem dos dados é um tópico Kafka que é desenhado para suportar filas de ingestão de dados, baixa latência e escalamento horizontal de processamento.

Ingestão de Dados com Apache Kafka

Devido ao volume de dados de milhões de mensagens por minuto, o tipo de armazenamento pelo Kafka precisa ser feito em um Bucket (Azure ou AWS) por motivo de latencia, custo e performance. É fundamental garantir que o Kafka seja configurado

corretamente e particionado adequadamente para otimizar a performance de processamento dos dados.
Além disso, a política de backup do bucket precisa ser definida para garantir que os dados salvos estejam seguros e sejam movidos para outros discos de armazenamento com custo menor.

Processamento em Tempo Real com Databricks e Apache Spark

O Databricks pode ser utilizado para processar os dados do Kafka utilizando o Databricks Structured Streaming que implementa os recursos do Apache Spark. É fundamental que se utilize os recursos de cache, indexação, otimização (como Liquid Clusters). Além de projetar corretamente as tabelas para otimizar os relacionamentos, e queries visando reduzir o custo computacional do processo.

Armazenamento e Consultas com Delta Lake

O Delta Lake do Databkicks será implementado com uma arquitetura de camadas: Bronze, Prata e Ouro. Os dados ingeridos pelo Databricks serão carregado as-is diretamente na camada Bronze.
As camadas seguintes, Prata e Ouro serão responsáveis por realizar o processamento, limpeza para criação do data mart com as informações refinadas para os relatórios.

A abordagem ELT (Extract, Load, Transform) será adotada para desacoplar os dados e garantir que as cópias originais sejam mantidas, o que, além de melhorar a governança, também reduz os custos e o tempo de processamento, simplificando a complexidade do processo de ingestão de dados.

Visualização e Relatórios

A criação dos relatórios irá consumir as tabelas com os cálculos refinados dos relatórios detalhados de notificação e Cobrança, que será realizada pelo Power BI, no entanto, Metabase, Superset, Qlik Sense também poderia realizar.

A escolha por utilizar o Power BI está no recurso de modelagem de conexão com serviços em nuvem, que permite otimizar os dados e armazená-los diretamente na nuvem do Power BI, eliminando a necessidade de um banco de dados para essa finalidade.

O resultado final o fluxo seguiria da seguinte forma

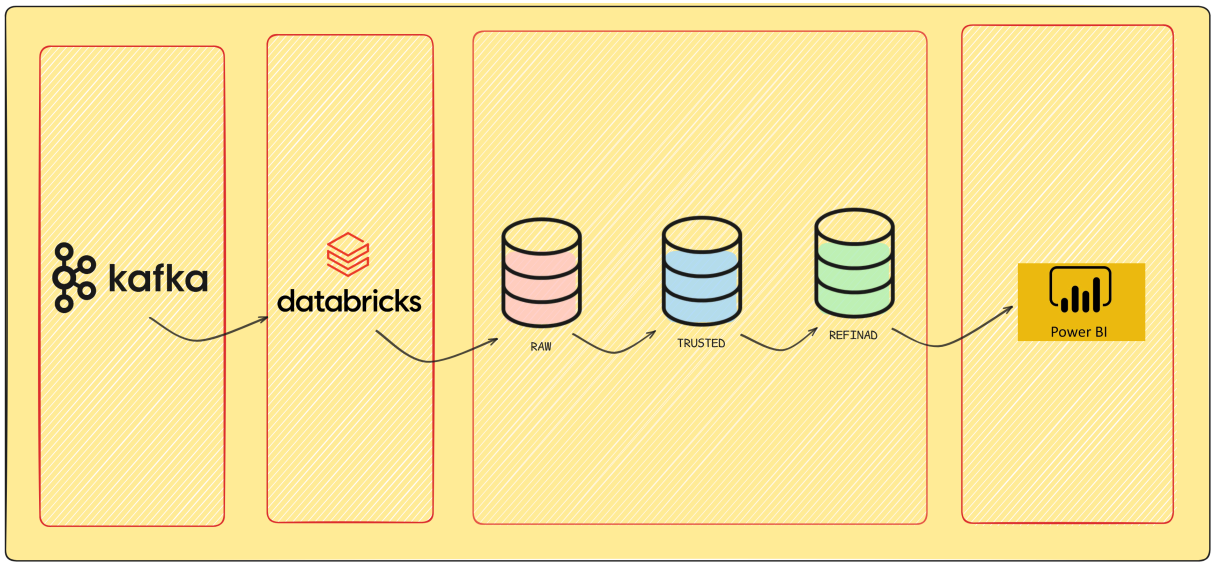

Figura 1 - Fluxograma das ferramentas

O resultado é um sistema enxuto, composto por ferramentas PaaS (Databricks e Power BI), que oferecem à equipe segurança e estabilidade no uso das plataformas. Os bancos de dados (buckets) são classificados como IaaS, garantindo alta disponibilidade e performance para a equipe. Um ponto de atenção importante é o orçamento do projeto. A estimativa de custos deve ser cuidadosamente avaliada para garantir que o projeto esteja alinhado com o orçamento disponível.

Questão 2:

Com base no arquivo invoices.csv, escreva um serviço que computa a média de faturamento de cada conta (account) nos últimos três e seis meses retroativos à Jan/2020. Quando não há dados suficientes na janela de 3 ou 6 meses o serviço registra a entrada como null.

Resultado esperado:

customer	account	avg_invoices_last_3_months	avg_invoices_last_6_months
C1000	A1100	56.333,33	41.416,66
C1000	A1200	50.666,66	null
C1000	A1300	null	null
C2000	A2100	55.400,00	null

Resposta da Questão 2

Resumo:

Foram criados dois scripts (Python e SQL) para realizar o calculo das faturas retroativas com as regras de negocios apresentadas.

A solução em python realiza o que foi solicitado, porém não oferece escalabilidade para operação, a transformação dos dados diretamente pode causar um custo alto de processamento dependendo do volume de dados, tem risco de desastres porque não tem armazenamento da transformação como em um banco de dados através dos recursos de ACID.

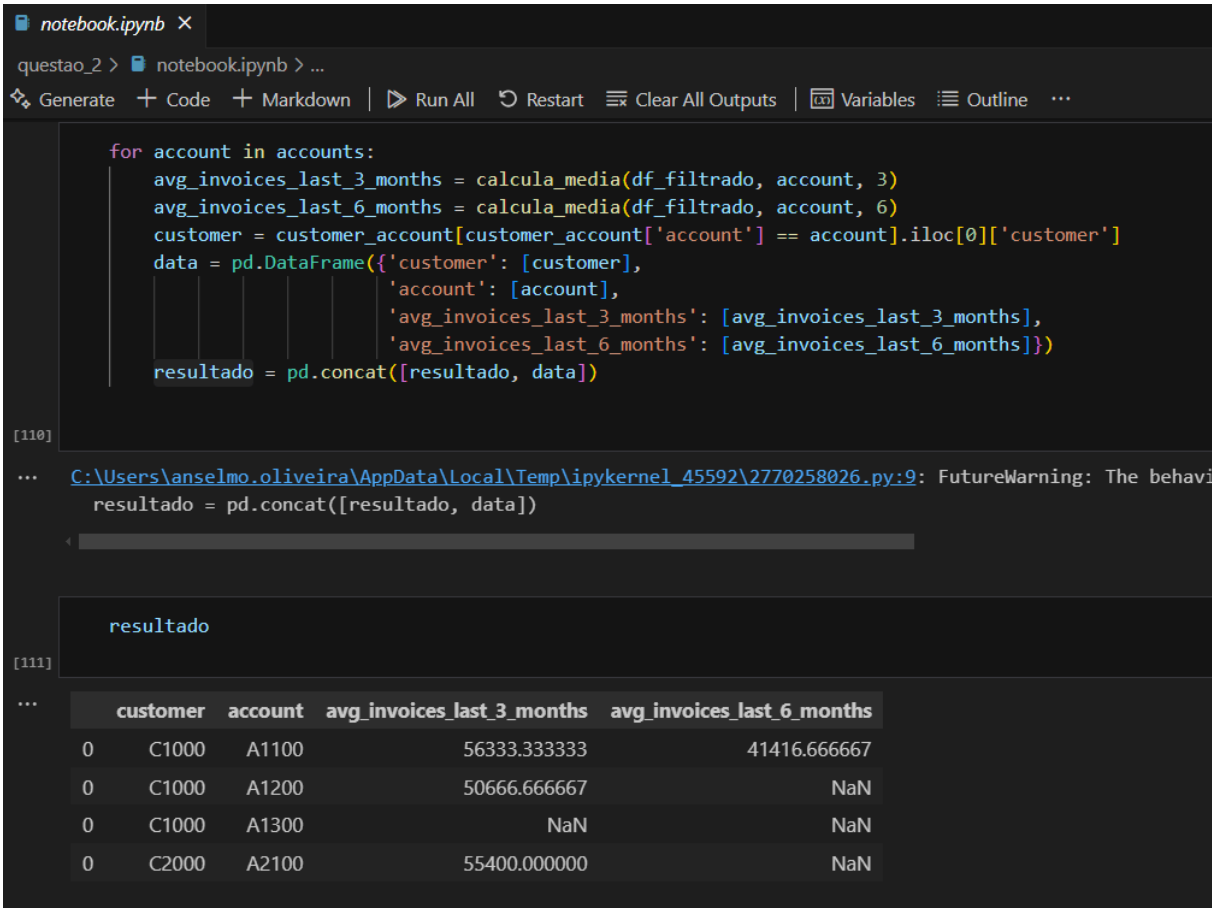

Figura 3 - Saída do programa Python

A segunda solução foi utilizada o banco de dados, onde os dados foram carregados full em uma tabela as-is e transformados utilizando subquery. Esse é um formato escalavel onde os dados podem ser salvos, gerenciados, documentados e protegidos e otimizados em um ambiente de produção.

Outro ponto importante é que realizar transformações em dados direto na linguagem é sensível a versão da biblioteca e linguagem. Por outro lado, nas soluções baseadas em bancos de dados, as queries geralmente mantêm sua sintaxe estável, com menos necessidade de ajustes após atualizações de versão.

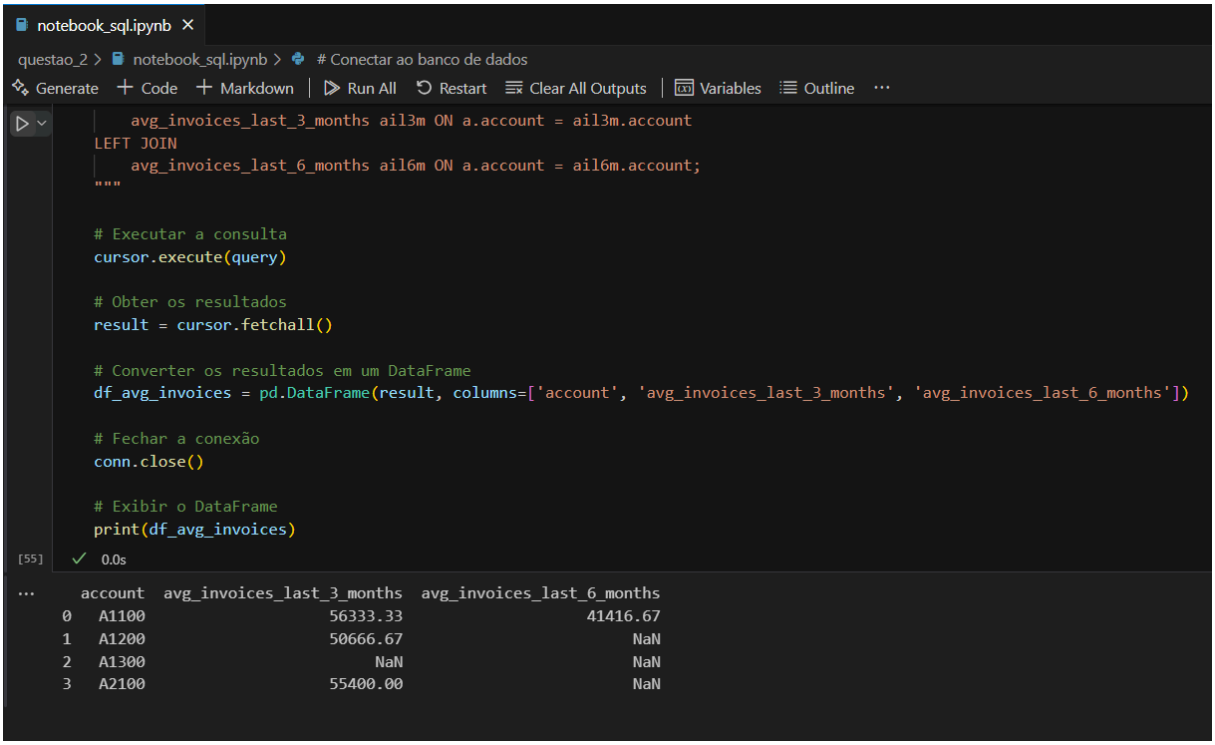

Figura 2 - Saída do programa SQL

Passo a passo da solução:

- Filtrar os registros para listar apenas os registros antes de 2020-01-01
- Salvar os dados no banco de dados (para a solução SQL)
- Contar quantos registros existem entre a data de referência, a data de referência - 3 meses, e a data de referência - 6 meses.
- Atribuir nulo para os registros que não atingirem a quantidade mínima e calcular a média para os dados com a quantidade minima.
- Construir um novo dataframe para armazenar os dados com customer e account e médias.

A saída da query obteve o resultado esperado.

Questão 3:

Uma plataforma de comunicação fornece fluxos de conversação (chatbots) entre outras funcionalidades. O data lake desta plataforma armazena valores informados pelos usuários em um formato semiestruturado (JSON) particionado por hora:

● hour=13.json
● hour=14.json

Considere que a seção content mantém as respostas de usuários. Ou seja, os valores preenchidos pelos usuários durante uma conversa. Para isto, ela armazena mapas onde a chave e valor são do tipo String.

Implemente um serviço que gera um relatório consolidado das últimas respostas informadas pelos usuários na hora 13 e hora 14 no seguinte formato:

customer	flow	session	first_answer_dt	last_answer_dt	name	cpf	delivery_confirmed
C1000	F1000	S1000	2019-12-16T13:59:58	2019-12-16T14:00:01	maria	305.584.960-40	true
C1000	F1000	S2000	2019-12-16T13:59:59	2019-12-16T14:00:00	joao	733.600.420-26	false

Os campos first_answer_dt e last_answer_dt representam, respectivamente, a primeira e última interações válidas (diferente de vazio). No exemplo acima, os campos name, cpf e delivery_confirmed são as respostas do usuário.

Para concluir, o serviço deve ser agnóstico de conversa, ou seja, deve suportar conteúdo de qualquer fluxo. Exemplo:

customer	flow	session	first_answer_dt	last_answer_dt	recomenda	nota
C2000	F2000	S3000	2019-12-16T13:59:59	2019-12-16T14:00:01	Simmmmmmm	9

Neste exemplo, recomenda e nota são as respostas do usuário.

Resposta da Questão 3

Como discutido no desafio anterior, a solução com persistência em banco de dados está mais alinhada às arquiteturas modernas de dados. Por isso, essa abordagem foi adotada na implementação.

Os dados recebidos se tratavam de dados semi-estruturados, que significa que seja em alguns casos, mais difícil armazenar em tabelas. Quando trabalhamos com dados semi-estruturados com cardinalidade de N-1 ou N-N, a melhor solução é a criação da modelagem de banco de dados, aplicando relacionamento nas tabelas.

Neste caso, os arquivos JSON possuíam cardinalidade 1-1, portanto, foram armazenados em uma tabela plana. É importante destacar que, nesse exemplo, cada fluxo (flow) terá as mesmas perguntas e os mesmos atributos, sendo assim os relatórios podem ser gerados respeitando registros do mesmo flow.

Passo a passo da solução:

- Fazer a leitura dos dados e fazer a ingesão no banco de dados.
- Utilizar a "ROW_NUMBER" para selecionar a ultima mensagem valída do do customer, flow e session.
- Remover os resultados nulos e strings vazias.
- Fazer uma subquery para selecionar a data de ínicio e fim da conversa para os mesmos customer, flow e session.
- Concatenar e ter a tabela final no formato desejado.
- Processar cada flow individualmente, pois cada um possui uma estrutura de tabela diferente.
- Acessar cada combinação possível de customer, flow, e session para obter todas as conversas.
- Efetuar a leitura das colunas customer, flow e session, first_answer_dt e last_answer_dt que serão iguais devido ao filtro por conversa, obtendo apenas a primeira linha.
- Transpor os registros de key e value, ordenados pela data de envio.
- Concatenar os dados e compor a tabela de saída como apresentada no desafio.

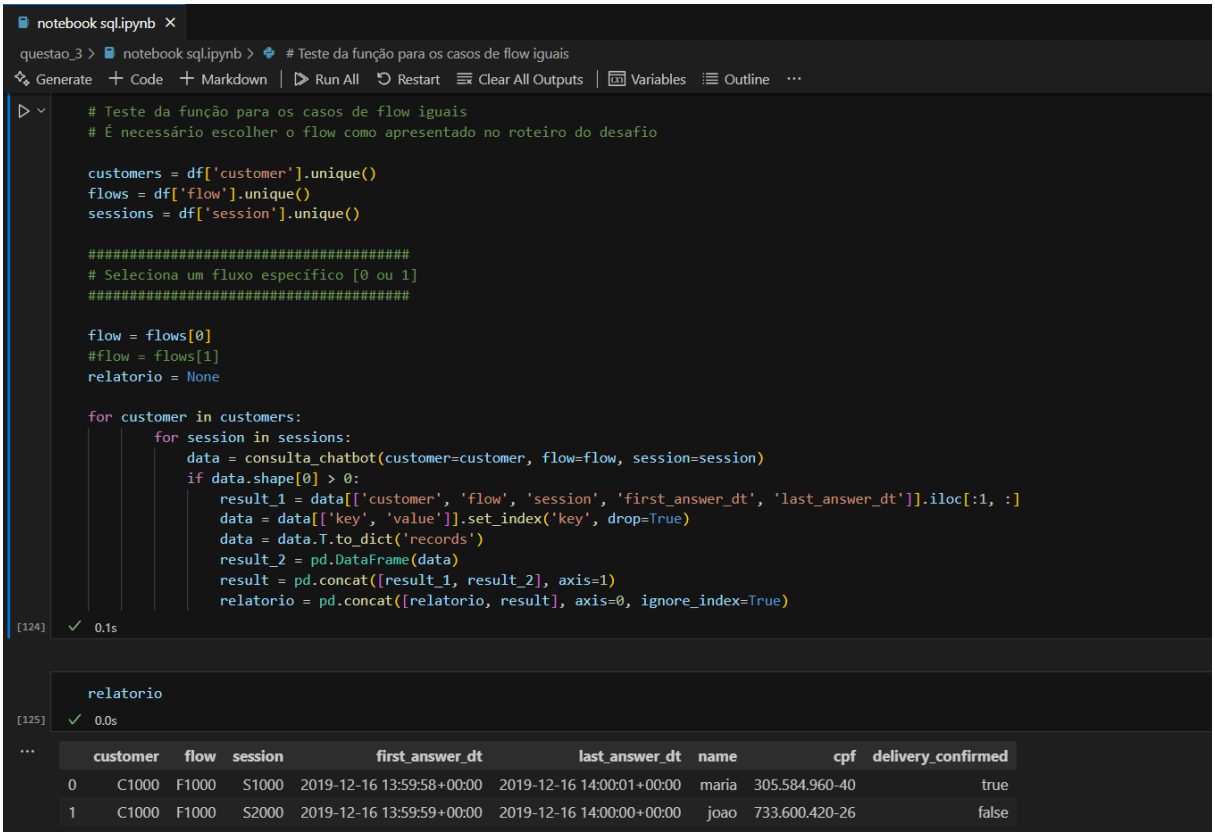

Figura 4 - Saída do programa para flow 1

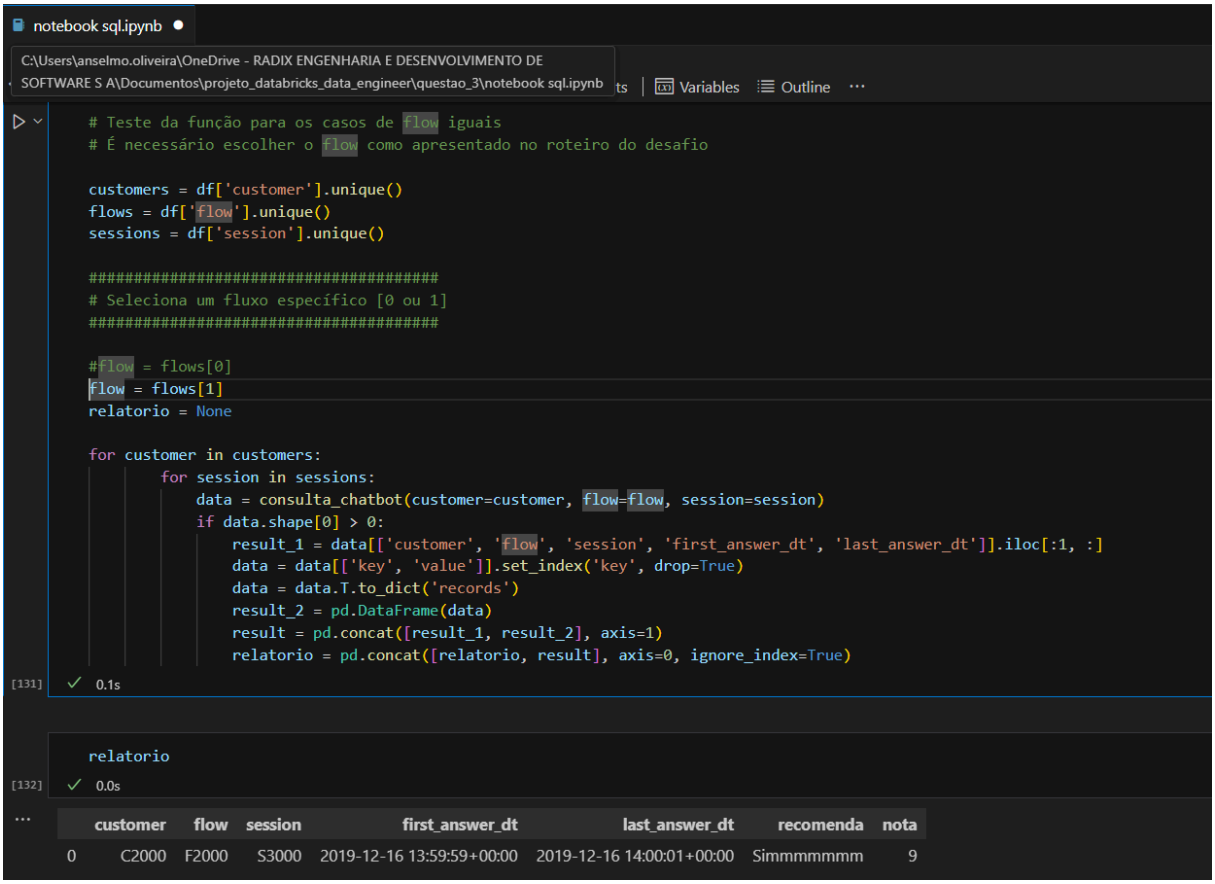

Figura 5 - Saída do programa para flow 2

A saída do relatório conforme a resposta apresentada.