Prova Equipe de Dados

Somos a CantuStore: Plataforma de tecnologia e logística que viabiliza soluções completas em pneus, guiando quem compra e apoiando quem vende. Se o assunto é pneu, você resolve aqui. Produtos e serviços em uma experiência 360° para abrir caminhos e ver pessoas e negócios evoluindo junto com a gente. Afinal, ficar parado não é opção, pelos menos pra nós.

Parte 1 - SQL

1.1 Campeonato

A organização e os resultados de um campeonato estão criados na seguinte estrutura:

```
CREATE TABLE times(
        time_id INTEGER NOT NULL
        ,time_nome  VARCHAR NOT NULL
        UNIQUE(time_id)
)
```

Time_id	Time_nome
10	Financeiro
20	Marketing
30	Logística
40	TI
50	Dados

```
 CREATE TABLE jogos (
        jogo_id INTEGER NOT NULL
        , mandante_time INTEGER NOT NULL
        , visitante_time INTEGER NOT NULL
        , mandante_gols INTEGER NOT NULL
        , visitante_gols INTEGER NOT NULL
        UNIQUE(jogo_id)
)
```

Jogo_id	Time_mandante	Time_visitante	Gols_mandante	Gols_Visitante
1	30	20	1	0
2	10	20	1	2
3	20	50	2	2
4	10	30	1	0
5	30	50	0	1

Calcule o número total de pontos que cada equipe marcou após todas as partidas, as regras são as seguintes:

- se uma equipe vencer uma partida (marcar mais gols que a outra equipe) ela ganha 3 pontos;

- se uma equipe empatar uma partida (marca exatamente o mesmo número de gols que a outra equipe) ganha um ponto;

- se uma equipe perder uma partida (marcar menos gols do que a outra equipe), ela não ganha pontos.

Escreva uma consulta que retorne uma classificação de todas as equipes (time_id) descritas na tabela times. Para cada equipe forneça seu nome e número de pontos que recebeu após todas as partidas (num_pontos). A tabela deve ser ordenada por num_pontos em ordem desc. Em caso de empate ordenar as linhas por time_id.

1.2 Comissões

A tabela Comissões tem a seguinte estrutura:

```
CREATE TABLE comissoes (
        comprador VARCHAR NOT NULL
        ,vendedor VARCHAR NOT NULL
        ,dataPgto DATE NOT NULL
        ,valor FLOAT NOT NULL
)
```

Escreva uma query em SQL que retorna a lista de vendedores que tem recebido até 1024 reais em até três transferências. Em outras palavras, o nome do vendedor vai ser listado se existir três ou menos comissões de modo que a soma dos valores recebidos não seja inferior a 1024 reais. Observe que pode haver mais de três comissões para mesmo vendedor, desde que três ou menos comissões totalizem pelo menos 1024 reais. O resultado por pode ser ordenado por nome do vendedor (ordem ascendente).

As colunas comprador e vendedor contém nomes dos correspondentes envolvidos nas comissões. A coluna data contém a data do pagamento da comissão, e a coluna valor contém o valor recebido em reais. Você pode assumir que o comprador é diferente do vendedor em cada linha.Por exemplo nesses dados:

Comprador	Vendedor	Data	Valor

Leonardo	Bruno	01/01/2000	200,00
Leonardo	Matheus	27/09/2003	1.024,00
Leonardo	Lucas	26/06/2006	512,00
Marcos	Lucas	17/12/2020	100,00
Marcos	Lucas	22/03/2002	10,00
Cinthia	Lucas	20/03/2021	500,00
Mateus	Bruno	02/06/2007	400,00
Mateus	Bruno	26/06/2006	400,00
Mateus	Bruno	26/06/2015	200,00

O vendedor Lucas foi listado porque ele recebeu R$ 1.112 reais em até três vendas: 512 + 100 + 500 = 1112. O vendedor Matheus foi listado porque recebeu R$ 1.024 em uma única transferência. O vendedor Bruno recebeu R$ 1.200 em quatro vendas, porém não foi listado porque em três vendas o total não chegou até R$ 1024.

O nome da coluna no retorno não importa.

1.3 Organização Empresarial

Existe uma empresa em que cada empregado recebe um salário e tem no máximo um chefe direto. Cada funcionário também pode ter vários chefes indiretos. O funcionário A é um chefe indireto do funcionário B se uma das seguintes condições for atendida:

+ funcionário A é chefe direto do funcionário B (cada chefe direto também é chefe indireto);

+ funcionário A é chefe indireto do chefe direto do funcionário B.

Em outras palavras, os chefes indiretos do funcionário B são: chefe direto de B, chefe direto do chefe direto de B, etc.

Por exemplo, dada a seguinte hierarquia:

Leonardo -> Bruno

Bruno -> Cinthia

Cinthia -> Mateus

Pedro -> Cinthia

Onde A -> B denota que B é um chefe direto de A, os chefes indiretos de Leonardo são Bruno, Cinthia e Mateus. Observe que Pedro não é o chefe indireto de Leonardo.

Dizemos que o funcionário A é mais baixo na hierarquia do que o funcionário B se A tem mais chefes indiretos do que o funcionário B. No exemplo acima, Leonardo tem três chefes indiretos e Pedro tem apenas dois chefes indiretos, então Leonardo é mais baixo na hierarquia do que Pedro

Você recebe uma tabela de funcionários com a seguinte estrutura:

```
CREATE TABLE colaboradores (

        id INTEGER NOT NULL
```

```
    , nome VARCHAR NOT NULL
    , salario INTEGER NOT NULL
    , lider_id INTEGER NOT NULL
    UNIQUE(id)
)
```

Cada registro na tabela funcionários representa um único funcionário. O id do chefe da coluna contém o id do chefe direto de cada funcionário. Se o empregado não tem chefe direto, esta coluna contém NULL no registro do funcionário. A coluna salário contém valores não negativos.

Você pode assumir que não há referências cíclicas, ou seja. empregado é o chefe indireto de si mesmo.

Para cada funcionário, gostaríamos de encontrar o chefe indireto de classificação mais baixa do funcionário na hierarquia que ganha pelo menos o dobro do funcionário.

Escreva uma consulta SQL que retorne uma tabela composta por duas colunas: id do funcionário, id do chefe, ordenado por id do funcionário. Cada funcionário deve aparecer nesta tabela. A tabela de resultados deve ser classificada em ordem crescente na identificação do funcionário. O id do chefe da coluna deve conter o chefe indireto de cada funcionário que preenche a condição acima. Se nenhum dos chefes indiretos do empregado ganha pelo menos o dobro do empregado, o id do chefe da coluna no registro que representa este funcionário deve conter NULL.

Por exemplo, para:

Empregados:

Id	**Nome**	**Salário**	**Lider_Id**
40	Helen	1500	50
50	Bruno	3000	10
10	Leonardo	4500	20
20	Marcos	10000	NULL
70	Mateus	1500	10
60	Cinthia	2000	70
30	Wilian	1501	50

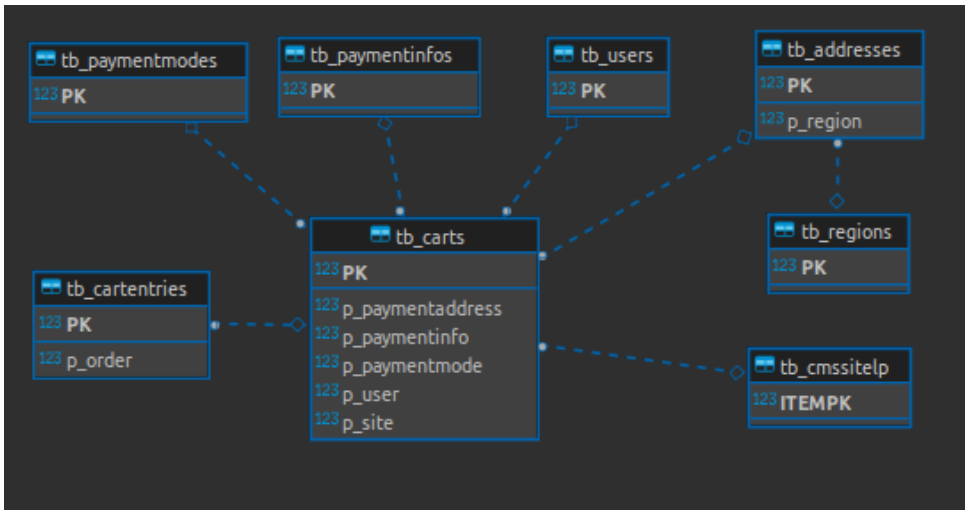
Parte 2 - Análise de Dados

Um dos problemas mais clássicos do mercado de e-commerce é o carrinho abandonado. Carrinho abandonado é aquele em que o cliente seleciona produtos para compra no e-commerce, mas não finaliza o processo. Algumas das estratégias para recuperá-lo são investir em remarketing, flexibilizar o frete e melhorar o checkout.

Analisar os dados desse evento é fundamental para que as empresas tentem entender o porquê que os clientes estão desistindo das compras.

Esse projeto visa ajudar a Cantu a coletar informações relacionadas a esse evento.

Considerando a seguinte estrutura e relacionamento dos dados que podem ser baixados nesse link:



Responda os seguintes questionamentos para ajudar a área responsável:

- Quais os produtos que mais tiveram carrinhos abandonados?

- Quais as duplas de produtos em conjunto que mais tiveram carrinhos abandonados?

- Quais produtos tiveram um aumento de abandono?

- Quais os produtos novos e a quantidade de carrinhos no seu primeiro mês de lançamento?

- Quais estados tiveram mais abandonos?

Além disso, para uma melhor avaliação da área:

- Gere um relatório dos produtos, mês a mês informando a quantidade de carrinhos abandonados, quantidade de itens abandonados e o valor não faturado;

- Gere também, um relatório por data informando a quantidade de carrinhos abandonados, quantidade de itens abandonados e o valor não faturado.

No final, exporte um arquivo .txt com os 50 carrinhos com os maiores carts.p_totalprice no seguinte layout:

```
carts.PK|carts.createdTS|carts.p_totalprice|user.p_uid|payment
modes.p_code|paymentinfos.p_installments|cmssitelp.p_name|addr
esses.p_postalcode|sum(cartentries.p_quantity)|count(cartentri
es.PK)

cartentries.p_product|cartentries.p_quantity|cartentries.p_tot
alprice|

cartentries.p_product|cartentries.p_quantity|cartentries.p_tot
alprice|
```

```
cartentries.p_product|cartentries.p_quantity|cartentries.p_tot
alprice|

cartentries.p_product|cartentries.p_quantity|cartentries.p_tot
alprice|

carts.PK|carts.createdTS|carts.p_totalprice|user.p_uid|payment
modes.p_code|paymentinfos.p_installments|cmssitelp.p_name|addr
esses.p_postalcode

cartentries.p_product|cartentries.p_quantity|cartentries.p_tot
alprice|

cartentries.p_product|cartentries.p_quantity|cartentries.p_tot
alprice|

carts.PK|carts.createdTS|carts.p_totalprice|user.p_uid|payment
modes.p_code|paymentinfos.p_installments|cmssitelp.p_name|addr
esses.p_postalcode

cartentries.p_product|cartentries.p_quantity|cartentries.p_tot
alprice|

cartentries.p_product|cartentries.p_quantity|cartentries.p_tot
alprice|

cartentries.p_product|cartentries.p_quantity|cartentries.p_tot
alprice|
```

Você pode utilizar Python e qualquer plataforma de Notebooks, mas, será um diferencial se você conseguir realizar toda a análise utilizando o Databricks e PySpark.