

Teste Técnico: Pessoa Engenheira de Dados Involves

1) Descreva com suas palavras os principais conceitos abaixo:

a) O que é um Data Warehouse ?

"Data Warehouses" (DW) ou "Armazém de Dados" são ambientes usados para armazenamento e gerenciamento de dados, utilizando o conceito de "OLAP" em detrimento do tradicional modelo OLTP de bancos transacionais, onde os dados são modelados a fim de reduzir possíveis redundâncias, erros e inconsistências, permitindo o processamento mais rápido de cálculos, funções ou qualquer transformação que visa extrair o máximo valor de dados anteriormente desestruturados. É um dos modelos possíveis de arquitetura de dados para utilização nas estruturas de inteligência de dados e um dos principais utilizados para confecção de relatórios e painéis.

b) Quais características possuem as tabelas do tipo Fato e Dimensão ?

Tabelas de Fato e Dimensões são características do modelo de arquitetura de data warehouses da primeira pergunta, onde os dados são modelados em **Dimensões**; tabelas que caracterizam as entidades relacionadas aos dados registrados, caracterizando-se por tabelas sem redundância e com valores exclusivos de uma entidade com seus respectivos IDs.

e **Fatos**; que são as observações, eventos, valores ou fatos relacionados aos dados registrados, onde esses são sumarizados e relacionados às respectivas dimensões através de conjuntos de chaves estrangeiras.

O relacionamento dessas tabelas forma os modelos chamados de "Modelo Estrela" ou "Snow Flake", dada a aparência visual que essa arquitetura gera.

c) O que é ETL ?

ETL é sigla para Extract, Transform and Load (Extração, Transformação e Carga). Trata-se de processos de engenharia de dados que visam movimentar, tratar e entregar/armazenar dados de acordo com sua necessidade, garantindo controle sobre o ciclo de vida dos dados e dos processos que envolvem a movimentação e transformação desses. É muito comum, também, o conceito de "ELT", onde a carga ou o armazenamento do dado é feito antes de qualquer transformação, garantindo o histórico de dados "brutos" da forma que foram gravados da fonte, permitindo que se verifique possíveis inconsistências provocadas diretamente na fonte, que podem ocorrer assim como nos processos posteriores de transformação, mantendo uma linearidade e fazendo-se possível investigar desvios em qualquer parte do processo. O ELT não anula a necessidade posterior de ETLs, sendo esses processos complementares.

d) Quais são as principais atribuições de um Engenheiro de Dados ?

O engenheiro de dados é o profissional responsável pela implementação dos processos de fluxo de dados (como o ETL/ELT) a fim de entregar os dados com integridade até o destino almejado. Esse profissional atua nas duas primeiras camadas do ciclo de vida de um dado, construindo e disponibilizando estruturas de dados brutos (Source of Raw - bronze) e dados tratados (Source of Truth - Silver), garantindo a ingestão e algumas transformações iniciais que garantam a veracidade

do dado a fim de facilitar o uso posterior por outros profissionais de dados e negócios. Esse profissional também é responsável pelo conceito de "Data Ops", garantindo infraestrutura e arquitetura necessária para que os processos de dados funcionem.

e) O que é Trade Marketing ?

Na minha concepção, trade marketing trata das estratégias de posicionamento de produtos nos pontos de vendas físicos, permitindo a promoção estratégica de produtos para seus respectivos públicos-alvo.

2) Crie uma query, considerando o SGBD MySQL, para exibir todos os dados de uma tabela de Pontos de Venda (tabela origem PONTO_VENDA_UNIDADE) e restringir apenas os pontos de venda que possuem sell in maior que 20.000 (campo SELLIN) e ainda ordená-los por nome do ponto de venda (campo NOME_PDV).

```
select *  
from PONTO_VENDA_UNIDADE  
where sellin > 20000  
order by NOME_PDV
```

3) Considerando a tabela de origem da questão anterior, crie uma query que some o valor de sell in de acordo com cada ponto de venda e agrupe os resultados por mês (campo MES) e ano (campo ANO). Ordene os registros por um período cronológico de forma crescente e por nome do ponto de venda.

```
select ANO, MES, NOME_PDV, sum(sellin) "valor_total"  
from PONTO_VENDA_UNIDADE  
group by ANO, MES, NOME_PDV  
order by ANO asc, MES asc, NOME_PDV
```

4) Considerando a tabela de origem da questão 2 e uma segunda tabela VISITAS_PONTO_VENDA, crie uma query que calcule a quantidade de visitas do ponto de venda de nome INVOLVES, sabendo-se que a tabela de visitas possui um campo que identifica se o ponto de venda foi visitado ou não chamado FL_VISITADO (Se 1 = Ponto de venda visitado / Se 0 = Ponto de venda não visitado). O campo chave que liga as duas tabelas é ID_PDV (na tabela PONTO_VENDA_UNIDADE) e FK_PDV (na tabela VISITAS_PONTO_VENDA). A query deve mostrar apenas as informações de nome do ponto de venda e quantidade de visitas realizadas.

```
select PVU.NOME_PDV  
      , sum(VPV.FL_VISITADO) "visitas"  
from PONTO_VENDA_UNIDADE PVU  
inner join VISITAS_PONTO_VENDA VPV  
on PVU.ID_PDV = VPV.FK_PDV  
where PVU.NOME_PDV = 'INVOLVES'  
group by PVU.NOME_PDV
```

5) Considerando a query abaixo, a pessoa engenheira de dados identificou que a performance da query está muito abaixo do esperado. Imaginando que um dos problemas possa estar relacionado aos índices das tabelas do banco de dados, a pessoa resolveu criar os índices nas tabelas. Liste quais possíveis campos devem ser indexados nas tabelas do banco de dados para que a query criada possa performar melhor. Leve em consideração que nenhum campo no banco de dados está indexado.

```
select
    FT.CICLO,
    FT.ID_DIM_PDV,
    FT.ID_BLOCO_ITEM,
    SUM(FT.QTD_PONTO_EXTRA),
    SUM(FTPI.TOTAL_NOTA_ITEM)
from FT_DOMINANCIA_PONTO_EXTRA_COMPLIANCE FT
inner join TABREF_PAINEL_LOJAS_LP TPLL
    on FT.ID_DIM_PDV = TPLL.ID_DIM_PDV
    and FT.CICLO = TPLL.CICLO
inner join FT_PERFECTSTORE_ITEM FTPI
    on FT.CICLO = FTPI.CICLO
    and FT.ID_DIM_PDV = FTPI.ID_DIM_PDV
    and FT.ID_BLOCO_ITEM = FTPI.ID_BLOCO_ITEM
    and FT.SEMANA_LP = FTPI.SEMANA_LP
where
    FT.CICLO = 202009
    and FT.ID_DIM_PDV = 223459792
group by FT.CICLO,
    FT.ID_DIM_PDV;
```

A princípio é possível observar dois campos de cada tabela que, se indexados, já fariam diferença na performance da consulta, sendo eles:

FT.CICLO, TPLL.CICLO, FTPI.CICLO: como o ciclo trata de período e possivelmente por conta disso aumenta a redundância das tabelas que possuem esse dado, a indexação do campo otimizaria a busca numa janela menor de tempo, aumentando a velocidade da consulta.

FT.SEMANA, FTPI.SEMANA: como no campo ciclo, deve haver uma redundância maior no campo semana por se tratar de um dado de período, o que, sendo verdade, também impactaria na performance da consulta em caso de indexação.

Aparentemente os outros campos não têm uma granularidade ou uma redundância que justifique a indexação, sendo que a indexação dos campos sugeridos já otimizaria o tempo da consulta. Caso o banco de dados fique demasiadamente grande, pode-se pensar em técnicas de processamento de Big Data fora do SGBD.

6) Considere a instrução Python a seguir:

```
x = [ print(i) for i in range(10) if i % 2 == 0 ]
```

Após a execução dessa instrução no Python , a variável “x” conterá qual valor.

A instrução irá imprimir os números pares contidos entre 0 e 9, resultando numa variável x como uma lista com valores vazios, já que a instrução printa os valores na definição. Para que a variável guarde a lista de valores, bastaria tirar o print da construção da lista de números.

7) Faça um script em Python que peça dois números e imprima a soma.

```
# input valores
num1 = float(input("Digite o primeiro valor: "))
num2 = float(input("Digite o segundo valor: "))

# soma valores
soma = num1 + num2

# resultado
print("A soma dos números é: ", soma)
```

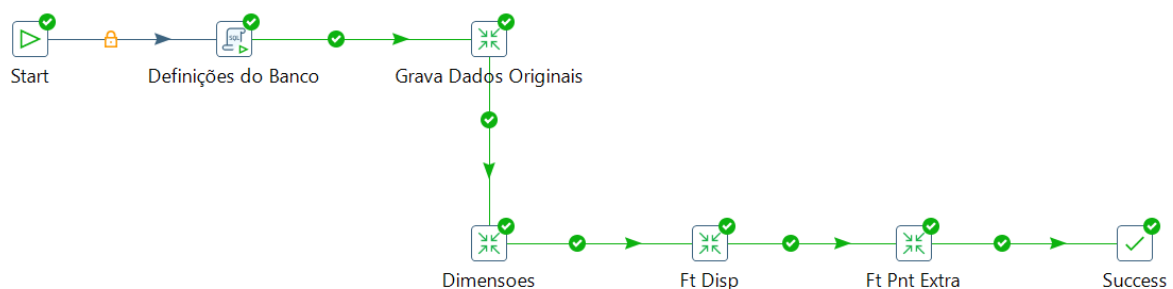
8, 9, 10)

Banco de dados Carga: MySQL Local

DB: testes

Orquestrador: Pentaho Data Integration

Repositório:



Tabelas:


DIM_CALENDARIO

```
SELECT `data`, mes, ano
FROM testes.dim_calendario;
```

<

calendario 1 ×

CT `data`, mes, ano FROM testes.dim_calendar

 data	¹²³ mes	¹²³ ano	
2020-09-01	9	2.020	
2020-09-02	9	2.020	
2020-09-03	9	2.020	
2020-09-04	9	2.020	
2020-09-05	9	2.020	
2020-09-06	9	2.020	
2020-09-07	9	2.020	
2020-09-08	9	2.020	
2020-09-09	9	2.020	
2020-09-10	9	2.020	
2020-09-11	9	2.020	
2020-09-12	9	2.020	
2020-09-13	9	2.020	
2020-09-14	9	2.020	
2020-09-15	9	2.020	
2020-09-16	9	2.020	
2020-09-17	9	2.020	
2020-09-18	9	2.020	
2020-09-19	9	2.020	
2020-09-20	9	2.020	

DIM_PDV

```
SELECT id, nome_ponto_venda, perfil_ponto_venda  
FROM testes.dim_pdv;
```

<

pdv 1 ×

CT id, nome_ponto_venda, perfil_ponto_venda FROM testes.dim_pdv

id	nome_ponto_venda	perfil_ponto_venda
115	INVOLVES	ATACADAO
116	INVOLVES STAGE	VAREJISTA

DIM_LINHA_PRODUTO

```
SELECT id, nome_linha_produto, perfil_linha_produto  
FROM testes.dim_linha_produto;
```

<

linha_produto 1 ×

CT id, nome_linha_produto, perfil_linha_produto FROM testes.dim_linha_pr

id	nome_linha_produto	perfil_linha_produto
398	BISCOITOS SORTIDOS	INVOLVES ZERO
407	MARGARINA	INVOLVES ZERO
408	MANTEIGA	INVOLVES ZERO
422	LEITE EM PO	INVOLVES ZERO
423	MAISENA	INVOLVES ZERO

FT_DISPONIBILIDADE

```
SELECT id_data, id_pdv, id_lp, disponibilidade  
FROM testes.ft_disponibilidade;
```

<

Disponibilidade 1 ×

CT id_data, id_pdv, id_lp, disponibilidade FROM testes.ft_disponibilidade

¹²³ id_data	¹²³ id_pdv	¹²³ id_lp	¹²³ disponibilidade
202.009	115	398	2
202.009	115	407	1
202.009	115	408	1
202.009	115	422	1
202.009	115	423	2
202.009	116	407	2
202.009	116	422	2

FT_DISPONIBILIDADE_AGREGADA

```
SELECT id_data, id_pdv, disponibilidade  
FROM testes.ft_disponibilidade_agregada;
```

<

Disponibilidade_agregada 1 ×

CT id_data, id_pdv, disponibilidade FROM testes.ft_disponibil

¹²³ id_data	¹²³ id_pdv	¹²³ disponibilidade
202.009	115	7
202.009	116	4

FT_PONTO_EXTRA

```
SELECT id_data, id_pdv, id_lp, pontos_extras  
FROM testes.ft_ponto_extra;
```

<

ft_ponto_extra 1 ×

SELECT id_data, id_pdv, id_lp, pontos_extras FROM testes.ft_ponto_extra

id_data	id_pdv	id_lp	pontos_extras
202.009	115	398	4
202.009	115	407	6
202.009	115	408	7
202.009	115	422	5
202.009	115	423	0
202.009	116	398	7
202.009	116	407	4
202.009	116	408	4
202.009	116	422	7
202.009	116	423	3

FT_PONTO_EXTRA_AGREGADA

```
SELECT id_data, id_pdv, pontos_extras  
FROM testes.ft_ponto_extra_agregada;
```

<

ft_ponto_extra_agregada 1 ×

SELECT id_data, id_pdv, pontos_extras FROM testes.ft_ponto_extra_agregada

id_data	id_pdv	pontos_extras
202.009	115	22
202.009	116	25