

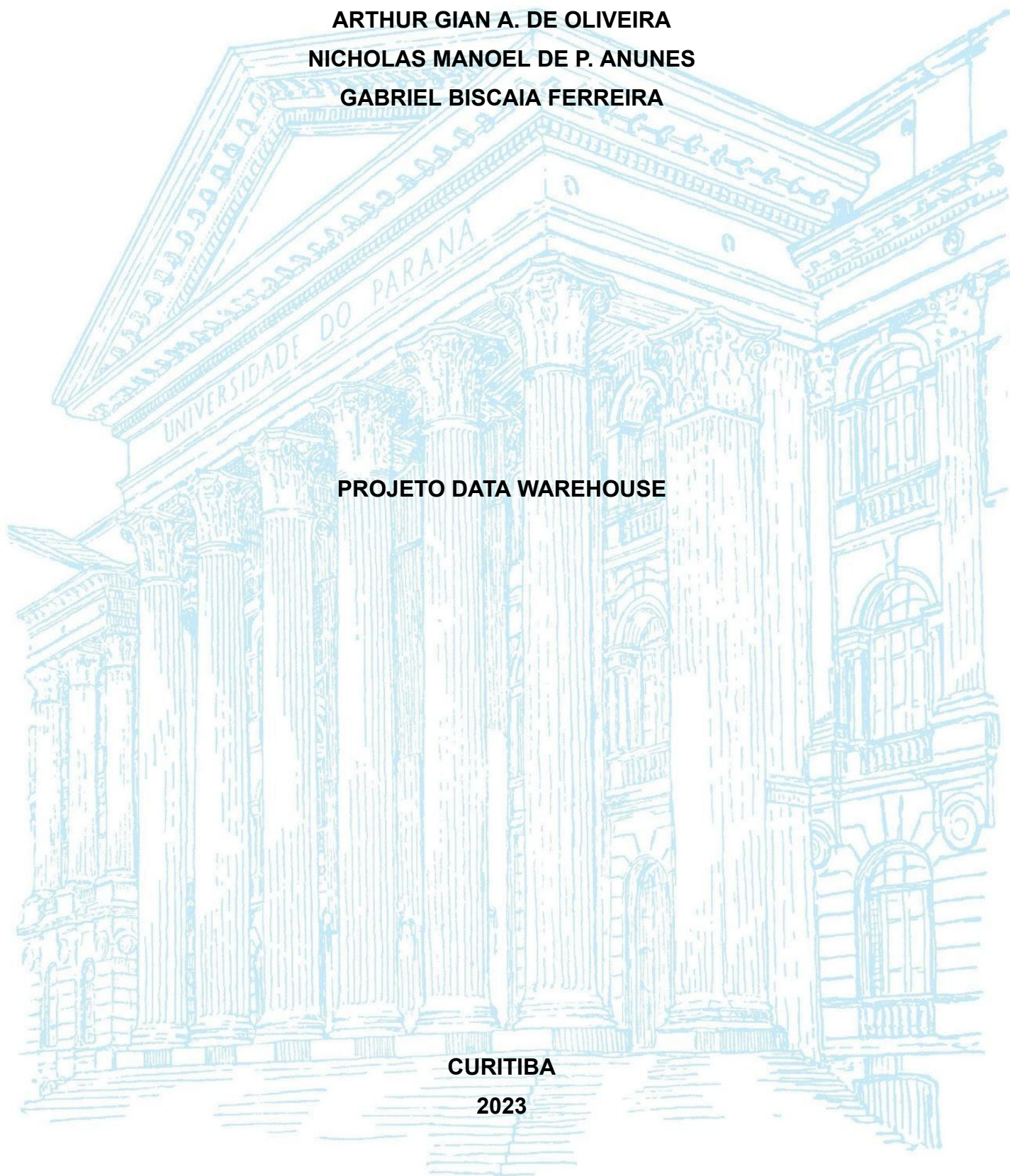
UNIVERSIDADE FEDERAL DO PARANÁ

**ALEXANDRE DIANO DE SOUZA
ARTHUR GIAN A. DE OLIVEIRA
NICHOLAS MANOEL DE P. ANUNES
GABRIEL BISCAIA FERREIRA**

PROJETO DATA WAREHOUSE

CURITIBA

2023



ALEXANDRE DIANO DE SOUZA
ARTHUR GIAN A. DE OLIVEIRA
NICHOLAS MANOEL DE P. ANUNES
GABRIEL BISCAIA FERREIRA

PROJETO DATA WAREHOUSE

Trabalho acadêmico apresentado ao curso de Graduação em Análise e Desenvolvimento de Sistemas, Universidade Federal do Paraná, referente ao seminário em equipe, como requisito parcial à obtenção de nota na disciplina de Banco de Dados III.

Professor(a): Prof. João Eugenio Marynowski

CURITIBA

2023

SUMÁRIO

1. INTRODUÇÃO	4
2. SITUAÇÃO E MODELO ATUAL	5
2.1. Diagrama Relacional - Projeto	5
2.2. Diagrama Relacional - Realidade	6
3. MODELO CONCEITUAL E LÓGICO DO BDM	7
3.1. Modelo Conceitual	7
3.1.1. Premissas	7
3.1.2. Diagrama Conceitual	8
3.2. Modelo Lógico do BDM	9
4. ARQUITETURA DO DW PROPOSTO	9
5. ETL	10
5.1. Migração de dados com Pentaho	10
5.2. Adaptação da SA	10
5.3. Criação das Tabelas Fato e Dimensões no DW	13
5.4. Inserção de Dados nas Tabelas Dimensão do DW	15
5.5. Inserção de Dados nas Tabelas Fato do DW	16
5.5.1. Inserção de Dados na Tabela Fato Compra	16
5.5.2. Inserção de Dados na Tabela Fato Venda	19
5.5.3. Inserção de Dados na Tabela Fato Lucro	22
6. ANÁLISE GERENCIAL	23
6.1. Quantidade de produtos vendidos por tipo e categoria ao longo do tempo	23
6.1.1. Evidência Análise Gerencial	24
6.2. Quantidade e valor total das compras por produto, possibilitando visão hierárquica ao longo do tempo	24
6.2.1. Evidências Análise Gerencial	28
6.2.1.1. Visão considerando produto, ano e mês	28
6.2.1.2. Visão considerando produto e mês	29
6.2.1.3. Visão considerando produto e ano	30
6.2.1.4. Visão considerando mês e ano	30
6.2.1.5. Visão considerando apenas mês	31
6.2.1.6. Visão considerando apenas ano	31
6.2.1.7. Visão considerando apenas produto	31
6.2.1.8. Visão sem considerar nada	32
6.3. Clientes que mais gastaram na loja virtual com quantidade acumulada, valor acumulado e média em um determinado período	32
6.3.1. Evidências Análise Gerencial	34
6.3.1.1. Período considerando dia, mês e ano	34
6.3.1.2. Período considerando mês e ano	35
6.3.1.3. Período considerando apenas ano	35

6.3.1.4. Período total	36
6.4. Últimas compras realizadas por cliente, e tempo decorrido até a data atual	36
6.4.1. Evidência Análise Gerencial	37
6.5. Lucratividade bruta dos produtos comprados e posteriormente vendidos	38
6.5.1. Evidências Análise Gerencial	38
6.5.1.1. Lucro por produto	38
6.5.1.2. Lucro total	39
6.6. Quantidade de atendimentos realizados ao longo do tempo.	39
6.6.1. Evidência Análise Gerencial	39
7. FREQUÊNCIA DE CARGA	39
8. TEMPO DE RETENÇÃO	40
9. HARDWARE E SGBDS RECOMENDADOS	40
10. CONSIDERAÇÕES FINAIS	40
11. REFERÊNCIAS	40
12. INFORMAÇÕES COMPLEMENTARES	41
12.1. Ferramentas Utilizadas	41
12.2. Evidências	41
12.2.1. Restauração do Arquivo de Backup no MS SQL Server	41
12.2.1.1. Importação Dados no MS SQL Server	42
12.2.1.2. Evidência da Execução do Script para Restaurar o Backup	43
12.2.1.3. Evidência da Base e Tabelas Criadas Após Restauração	44
ANEXOS	45

1. INTRODUÇÃO

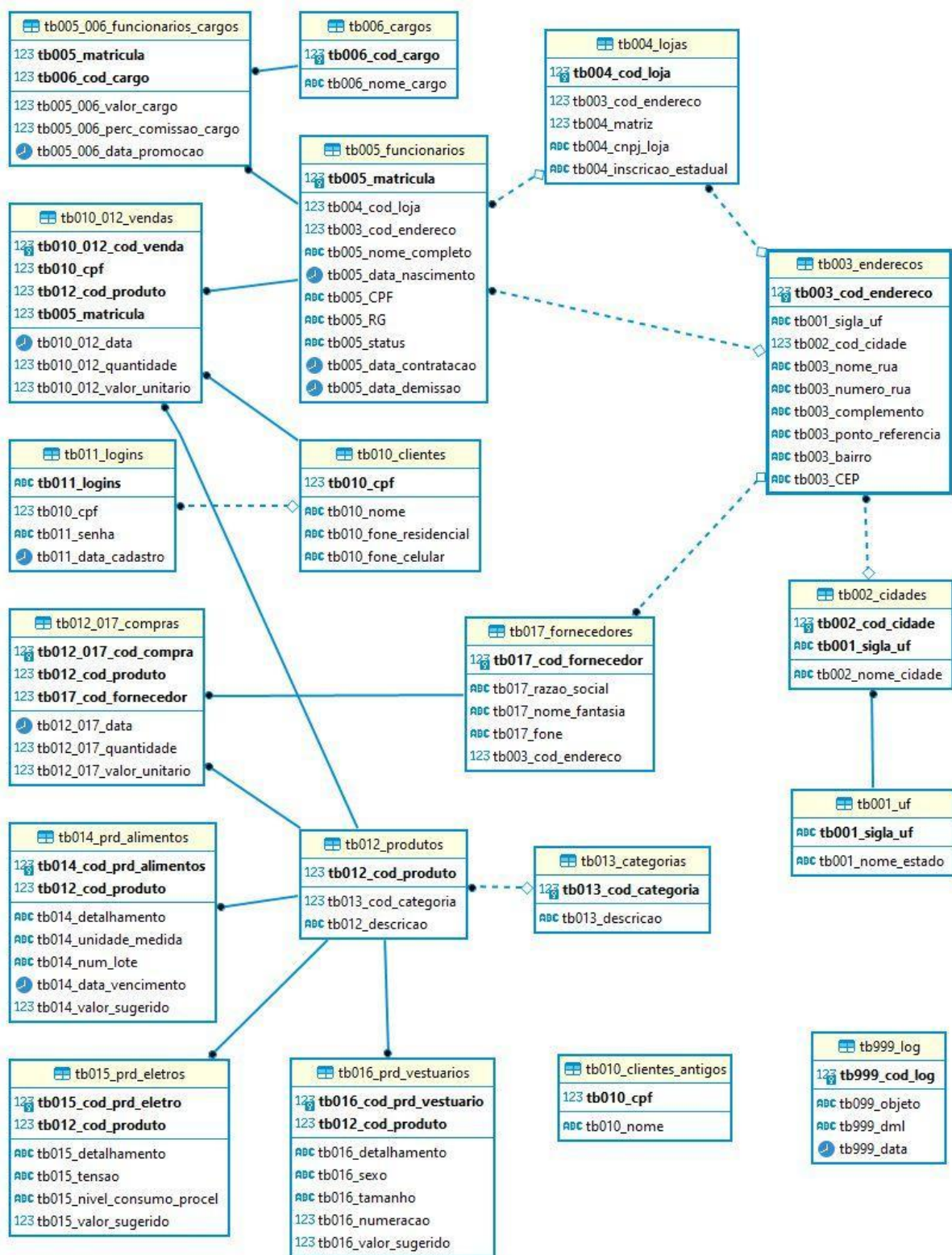
No contexto atual do mundo, com o avanço da complexidade das organizações junto com a ascensão tecnológica, a relevância da gestão dos repositórios de dados aumentou. Tomar decisões precisas em um mercado cada vez mais competitivo tornou-se crucial para a sobrevivência de uma organização.

Dessa forma, a Ciência de Dados desempenha um papel vital neste cenário, pois permite transformar uma grande quantidade de dados brutos em informação e conhecimento. A implantação de um Data Warehouse (DW) tem como objetivo abordar essa necessidade, reestruturando um backup de um banco de dados operacional de forma a tornar mais eficiente a extração de informações gerenciais com base em fatos.

Na situação apresentada, temos um banco de dados relacional operacional de uma cadeia de lojas de varejo, onde a análise das informações é extremamente demorada, prejudicando efetivamente as operações. Nesse contexto, a criação de um DW a partir desse banco de dados pode fornecer uma fonte de informações gerenciais sem os problemas atuais (processamento e extração de dados lentos), desde que seja utilizada uma infraestrutura apropriada. Atualmente, existem inúmeras opções, ferramentas e soluções de DW disponíveis no mercado com essa infraestrutura, tanto com implementação local quanto em nuvem, como o Amazon Redshift, o Microsoft Azure, entre outras, e cada uma delas possui características adequadas para diferentes aplicações de DW.

A proposta da nossa equipe é configurar uma instância de MS SQL Server e realizar a restauração do arquivo de backup BD_VAREJO.sql. Usaremos o DBeaver para conectar-se à base de dados da instância do MS SQL Server e criar o diagrama lógico relacional das tabelas com a finalidade de comparar os relacionamentos das tabelas com o diagrama do arquivo BD_VAREJO.png. Assim que o backup estiver restaurado no MS SQL Server, a proposta é transferir todos os dados do MS SQL Server para uma área de staging em um banco de dados PostgreSQL.

Após a transferência dos dados para a área de staging, o DW será desenvolvido no PostgreSQL a partir dos dados carregados na área de staging. A ferramenta Pentaho será usada para realizar o processo de ETL.



Fonte: Os Autores (2023)

2.3. Comparação Entre os Diagramas Real e de Projeto

Comparando-se o diagrama relacional do projeto com o diagrama relacional de fato implementado no BD físico, constatou-se que:

- Não existe a tabela tb018_itens_vendas;
- Não existe a tabela tb019_itens_compras;
- Tabela tb012_017_compras possui:
 - Chave estrangeira tb012_cod_produto;
 - Coluna tb012_017_quantidade;
 - Coluna tb012_017_valor_unitario.
- Tabela tb012_017_compras não possui:
 - Coluna tb012_017_forma_pagamento;
- Tabela tb019_012_vendas possui:
 - Chave estrangeira tb012_cod_produto;
 - Coluna tb010_012_quantidade;
 - Coluna tb010_012_valor_unitario;
- Tabela tb019_012_vendas não possui:
 - Coluna tb010_012_forma_pagamento;
- Tabela tb012_017_compras não possui relacionamento com tabela
- tb019_itens_compras, mas sim com a tabela tb012_produtos;
- Tabela tb010_tb012_vendas não possui relacionamento com a tabela
- tb018_itens_venda;
- Tabela tb012_produtos se relaciona com a tabela tb019_012_vendas;
- Tabela tb014_prd_alimentos possui uma coluna tb014_valor_sugerido;
- Tabela tb015_prd_eletros possui uma coluna tb015_valor_sugerido;
- Tabela tb016_prd_vestuarios possui uma coluna tb016_valor_sugerido;

3. MODELO CONCEITUAL E LÓGICO DO BDM

3.1. Modelo Conceitual

3.1.1. Premissas

Tipos:

- Alimentos;
- Eletros;
- Vestuário.

Categorias:

- Alimentos Perecíveis;
- Alimentos Não Perecíveis;
- Eletrodomésticos;
- Eletrônicos;
- CD e DVD;
- Roupas Unisex;
- Roupas Infantis.

Analisando o repositório de informações, observou-se que todos os clientes estão registrados. Consequentemente, tornou-se desafiante distinguir entre uma transação realizada virtualmente ou fisicamente. Portanto, assumimos como suposição que todas as transações ocorrem virtualmente e que todas as compras são efetuadas em plataformas digitais.

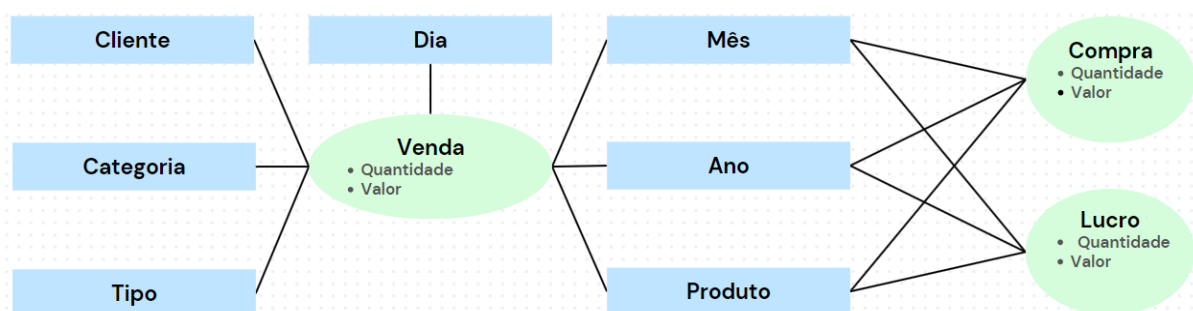
Chegamos à conclusão de que a dimensão temporal, especificamente o "dia," não possui a mesma relevância para a categoria "Compra" em comparação com a categoria "Venda." Portanto, optamos por eliminar a dimensão "dia" associada à categoria "Compra."

Consideramos que as vendas representam o único tipo de interação com os clientes. Portanto, a questão de gestão relacionada à "Quantidade de interações realizadas ao longo do período" se traduz na quantidade de vendas efetuadas durante esse período.

3.1.2. Diagrama Conceitual

Decidimos criar um diagrama seguindo o modelo Constelação, levando em consideração que a estrutura disponível nos permite ter dois eventos, Venda e Compra, ambos compartilhando características (mês, ano e produto). Além disso, temos um evento Lucro, que também compartilha as mesmas características de produto, mês e ano, acompanhado pelo atributo "valor" conforme requerido para as análises de gestão. Como ilustrado na Figura 3, Venda, Compra e Lucro apresentam configurações do tipo constelação, interconectadas através das dimensões em comum, formando assim uma estrutura de constelação.

Figura 3 - Diagrama Conceitual

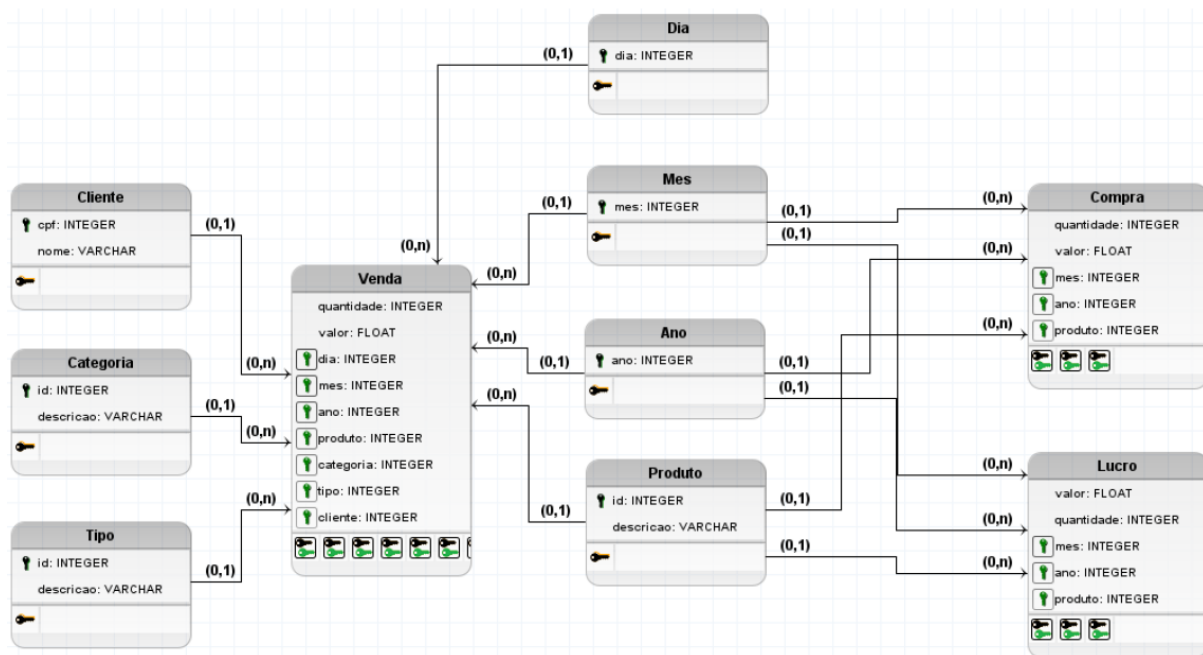


Fonte: Os Autores (2023)

3.2. Modelo Lógico do BDM

Com base no modelo conceitual previamente formulado, o modelo prático de concretização do Data Warehouse foi concebido. Neste esquema, foram estabelecidas as conexões e suas quantidades, além dos formatos de dados, chaves principais e chaves secundárias.

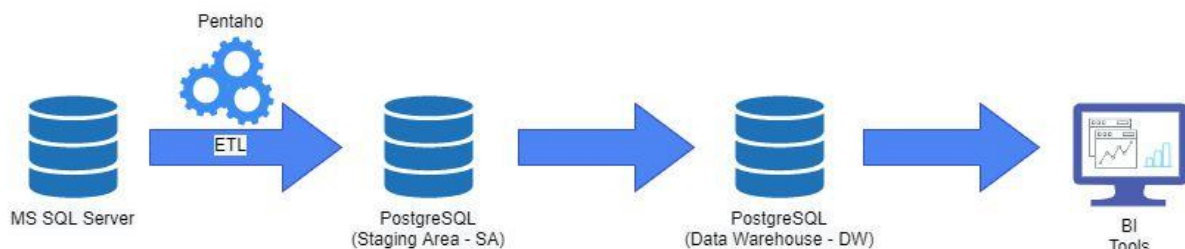
Figura 4 - Modelo Lógico



Fonte: Os Autores (2023)

4. ARQUITETURA DO DW PROPOSTO

Podemos observar na Figura 5 o modelo da arquitetura, onde foi utilizada a ferramenta Pentaho para realizar a extração dos dados da base relacional operacional (MS SQL Server) e inserção dos dados extraídos em uma Staging Area (SA) relacional (PostgreSQL). Uma vez os dados carregados na SA foram utilizadas queries SQL para a realização da extração e estruturação dos dados da SA e alimentação do DW.



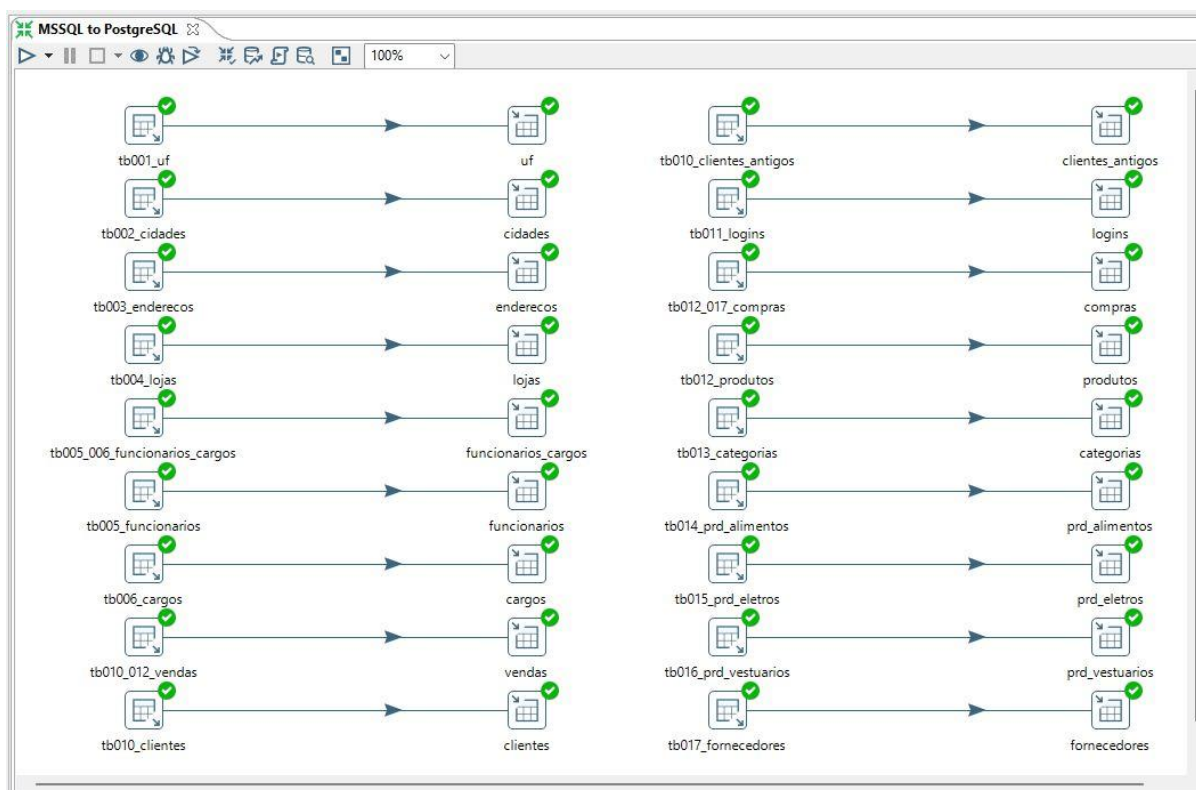
Fonte: Os Autores (2023)

5. ETL

5.1. Migração de dados com Pentaho

Antes da execução das ETLs para a alimentação do DW, realizamos a migração dos dados através do Pentaho para a nossa Staging Area (Figura 6), visando facilitar a aplicação das ETLs e geração dos Datamarts posteriormente.

Figura 6 - Migração com Pentaho



Fonte: Os Autores (2023)

5.2. Adaptação da SA

Vale ressaltar que na Staging Area foi adicionada uma coluna relacionada com o tipo do produto, com o intuito de auxiliar nas Análises Gerenciais.

```
// Adição da coluna tipo_produto na tabela produtos de staging
area
ALTER TABLE staging.produtos add tipo_produto int;

// Update da coluna tipo_produto na staging area conforme tipo de
cada produto
// Considerando produtos do tipo alimento
UPDATE staging.produtos
```

```

SET tipo_produto = 1
WHERE tb012_cod_produto IN (SELECT p.tb012_cod_produto FROM
staging.prd_alimentos pa, staging.produtos p
WHERE pa.tb012_cod_produto = p.tb012_cod_produto);

// Considerando produtos do tipo eletro
UPDATE staging.produtos
SET tipo_produto = 2
WHERE tb012_cod_produto IN (SELECT p.tb012_cod_produto FROM
staging.prd_eletros pa, staging.produtos p
WHERE pa.tb012_cod_produto = p.tb012_cod_produto);

// Considerando produtos do tipo vestuario
UPDATE staging.produtos
SET tipo_produto = 3
WHERE tb012_cod_produto IN (SELECT p.tb012_cod_produto FROM
staging.prd_vestuarios pa, staging.produtos p
WHERE pa.tb012_cod_produto = p.tb012_cod_produto);

```

Dada a necessidade de obter a Lucratividade na análise gerencial, tangenciamos a alternativa de realizar os cálculos através das médias de valor de compra e média de valor de venda, dessa maneira esse processamento teria que ocorrer no momento do ETL, com a criação de duas novas colunas na tabela Produto com os valores médios que serão utilizados posteriormente no DW para a realização do cálculo da lucratividade final.

```

// Adição da coluna preco_medio_compra na tabela produtos de
staging area
ALTER TABLE staging.produtos add preco_medio_compra float;

// Adição da coluna preco_medio_venda na tabela produtos de
staging area
ALTER TABLE staging.produtos add preco_medio_venda float;

// Cálculo da média ponderada do valor pago pelo produto na
compra e inserção
dos valores na coluna preco_medio_compra da tabela
staging.produtos
UPDATE staging.produtos p
SET preco_medio_compra = subquery.preco_medio_compra
FROM (
SELECT
p.tb012_cod_produto,

```

```

        p.tb012_descricao,
        sum((c.tb012_017_quantidade*c.tb012_017_valor_unitario))/sum(c.tb
012_017_quanti
        dade) AS preco_medio_compra
    FROM
        staging.compras c,
        staging.produtos p
    GROUP BY
        p.tb012_cod_produto,
        c.tb012_cod_produto,
        p.tb012_descricao
    HAVING
        c.tb012_cod_produto = p.tb012_cod_produto
    ) AS subquery
    WHERE
        p.tb012_cod_produto = subquery.tb012_cod_produto;

    // Cálculo da média ponderada do valor pago pelo produto na venda
e inserção
    dos valores na coluna preco_medio_venda da tabela
staging.produtos
    UPDATE staging.produtos p
    SET preco_medio_venda = subquery.preco_medio_venda
    FROM (
        SELECT
            p.tb012_cod_produto,
            p.tb012_descricao,
            sum((v.tb010_012_quantidade
            *v.tb010_012_valor_unitario))/sum(v.tb010_012_quantidade) AS
preco_medio_venda
        FROM
            staging.vendas v,
            staging.produtos p
        GROUP BY
            p.tb012_cod_produto,
            v.tb012_cod_produto,
            p.tb012_descricao
        HAVING
            v.tb012_cod_produto = p.tb012_cod_produto
        ) AS subquery
    WHERE
        p.tb012_cod_produto = subquery.tb012_cod_produto;

```

5.3. Criação das Tabelas Fato e Dimensões no DW

A seguir são apresentados os SQL de criação das tabelas fato e das tabelas dimensões, bem como os alter tables relativos à aplicação das chaves estrangeiras nas tabelas fato.

```
// Primeiro Passo
// Criar o schema e as tabelas
```

```
CREATE SCHEMA dw;
CREATE TABLE dw.Cliente (
    cpf BIGINT PRIMARY KEY,
    nome VARCHAR
);
CREATE TABLE dw.Ano (
    ano INTEGER PRIMARY KEY
);
CREATE TABLE dw.Mes (
    mes INTEGER PRIMARY KEY
);
CREATE TABLE dw.Produto (
    id INTEGER PRIMARY KEY,
    descricao VARCHAR,
    preco_medio_compra FLOAT,
    preco_medio_venda FLOAT
);
CREATE TABLE dw.Categoria (
    id INTEGER PRIMARY KEY,
    descricao VARCHAR
);
CREATE TABLE dw.Dia (
    dia INTEGER PRIMARY KEY
);
CREATE TABLE dw.Tipo (
    id INTEGER PRIMARY KEY,
    descricao VARCHAR
);
CREATE TABLE dw.Venda (
    quantidade INTEGER,
    valor FLOAT,
    dia INTEGER,
    mes INTEGER,
    ano INTEGER,
```



```

    produto INTEGER,
    categoria INTEGER,
    tipo INTEGER,
    cliente BIGINT
);
CREATE TABLE dw.Compra (
    quantidade INTEGER,
    valor FLOAT,
    mes INTEGER,
    ano INTEGER,
    produto INTEGER
);
CREATE TABLE dw.Lucro (
    produto INTEGER,
    quantidade INTEGER,
    lucro FLOAT
);

// Segundo Passo
// Adicionar as constraints
ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaCliente
    FOREIGN KEY (cliente)
    REFERENCES dw.Cliente (cpf);
ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaDia
    FOREIGN KEY (dia)
    REFERENCES dw.Dia (dia);
ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaMes
    FOREIGN KEY (mes)
    REFERENCES dw.Mes (mes);
ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaAno
    FOREIGN KEY (ano)
    REFERENCES dw.Ano (ano);
ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaProduto
    FOREIGN KEY (produto)
    REFERENCES dw.Produto (id);
ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaCategoria
    FOREIGN KEY (categoria)
    REFERENCES dw.Categoria (id);
ALTER TABLE dw.Venda ADD CONSTRAINT fkVendaTipo
    FOREIGN KEY (tipo)
    REFERENCES dw.Tipo (id);
ALTER TABLE dw.Compra ADD CONSTRAINT fkCompraMes
    FOREIGN KEY (mes)

```

```

REFERENCES dw.Mes (mes);
ALTER TABLE dw.Compra ADD CONSTRAINT fkCompraAno
FOREIGN KEY (ano)
REFERENCES dw.Ano (ano);
ALTER TABLE dw.Compra ADD CONSTRAINT fkCompraProduto
FOREIGN KEY (produto)
REFERENCES dw.Produto (id);
ALTER TABLE dw.Lucro ADD CONSTRAINT fkLucroProduto
FOREIGN KEY (produto)
REFERENCES dw.Produto (id);

```

5.4. Inserção de Dados nas Tabelas Dimensão do DW

Os SQL apresentados a seguir são referentes a inserção dos dados nas tabelas Dimensão do DW. Os dados inseridos nestas tabelas do DW foram obtidos a partir da SA, mediante processo de ETL.

```

// Inserção de dados na tabela dimensão Ano
INSERT INTO dw.ano
SELECT DISTINCT EXTRACT(YEAR FROM tb010_012_data) FROM staging.vendas
ON CONFLICT DO NOTHING;
INSERT INTO dw.ano
SELECT DISTINCT EXTRACT(YEAR FROM tb012_017_data) FROM
staging.compras
ON CONFLICT DO NOTHING;

// Inserção de dados na tabela dimensão Mes
INSERT INTO dw.mes
SELECT DISTINCT EXTRACT(MONTH FROM tb010_012_data) FROM
staging.vendas
ON CONFLICT DO NOTHING;
INSERT INTO dw.mes
SELECT DISTINCT EXTRACT(MONTH FROM tb012_017_data) FROM
staging.compras
ON CONFLICT DO NOTHING;

// Inserção de dados na tabela dimensão Dia
INSERT INTO dw.dia
SELECT DISTINCT EXTRACT(DAY FROM tb010_012_data) FROM staging.vendas
ON CONFLICT DO NOTHING;

// Inserção de dados na tabela dimensão Produto
INSERT INTO dw.produto
SELECT DISTINCT p.tb012_cod_produto, p.tb012_descricao,

```

```

p.preco_medio_compra, p.preco_medio_venda FROM staging.produtos p;

// Inserção de dados na tabela dimensão Cliente
INSERT INTO dw.cliente
  SELECT DISTINCT c.tb010_cpf, c.tb010_nome FROM staging.clientes c;

// Inserção de dados na tabela dimensão Categoria
INSERT INTO dw.categoria
  SELECT DISTINCT c.tb013_cod_categoria, c.tb013_descricao FROM
staging.categorias c;

// Inserção de dados na tabela dimensão Tipo
INSERT INTO dw.tipo (id, descricao)
  VALUES (1, 'alimento'), (2, 'eletro'), (3, 'vestuario');

```

5.5. Inserção de Dados nas Tabelas Fato do DW

Os SQL apresentados a seguir são referentes a inserção dos dados nas tabelas Fato do DW. Os dados inseridos nestas tabelas do DW foram obtidos a partir da SA, mediante processo de ETL.

5.5.1. Inserção de Dados na Tabela Fato Compra

```

// Inserção de dados na tabela fato Compra
INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
  SELECT
    c.tb012_cod_produto AS produto,
    c.tb012_017_quantidade AS quantidade,
    c.tb012_017_quantidade*c.tb012_017_valor_unitario AS valor,
    EXTRACT(MONTH FROM c.tb012_017_data) AS mes,
    EXTRACT(YEAR FROM c.tb012_017_data) AS ano
  FROM
    staging.compras c
);

// Independente do mês
INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
  SELECT
    c.tb012_cod_produto AS produto,
    SUM(c.tb012_017_quantidade) AS quantidade,
    SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
    NULL AS mes,
    EXTRACT(YEAR FROM c.tb012_017_data) AS ano

```

```

FROM
    staging.compras c
GROUP BY c.tb012_cod_produto, EXTRACT(YEAR FROM c.tb012_017_data)
);

// Independente do ano
INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
    SELECT
        c.tb012_cod_produto AS produto,
        SUM(c.tb012_017_quantidade) AS quantidade,
        SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
        EXTRACT(MONTH FROM c.tb012_017_data) AS mes,
        NULL AS ano
    FROM
        staging.compras c
    GROUP BY c.tb012_cod_produto, EXTRACT(MONTH FROM c.tb012_017_data)
);

// Independente do ano e mês
INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
    SELECT
        c.tb012_cod_produto AS produto,
        SUM(c.tb012_017_quantidade) AS quantidade,
        SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
        NULL AS mes,
        NULL AS ano
    FROM
        staging.compras c
    GROUP BY c.tb012_cod_produto
);

// Independente de produto
INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
    SELECT
        NULL AS produto,
        SUM(c.tb012_017_quantidade) AS quantidade,
        SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
        EXTRACT(MONTH FROM c.tb012_017_data) AS mes,
        EXTRACT(YEAR FROM c.tb012_017_data) AS ano
    FROM

```

```

        staging.compras c
    GROUP BY EXTRACT(MONTH FROM c.tb012_017_data), EXTRACT(YEAR FROM
c.tb012_017_data)
);

// Independente do produto e mês
INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
    SELECT
        NULL AS produto,
        SUM(c.tb012_017_quantidade) AS quantidade,
        SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
        NULL AS mes,
        EXTRACT(YEAR FROM c.tb012_017_data) AS ano
    FROM
        staging.compras c
    GROUP BY EXTRACT(YEAR FROM c.tb012_017_data)
);

// Independente do produto e ano
INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
    SELECT
        NULL AS produto,
        SUM(c.tb012_017_quantidade) AS quantidade,
        SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
        EXTRACT(MONTH FROM c.tb012_017_data) AS mes,
        NULL AS ano
    FROM
        staging.compras c
    GROUP BY EXTRACT(MONTH FROM c.tb012_017_data)
);

// Independente do produto, mês e ano
INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
    SELECT
        NULL AS produto,
        SUM(c.tb012_017_quantidade) AS quantidade,
        SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
        NULL AS mes,
        NULL AS ano
    FROM

```

```
    staging.compras c
);
```

5.5.2. Inserção de Dados na Tabela Fato Venda

```
// Inserção de dados na tabela fato Venda
INSERT INTO
dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo,
cliente)
(
    SELECT
        v.tb010_012_quantidade AS quantidade,
        v.tb010_012_quantidade*v.tb010_012_valor_unitario AS valor,
        EXTRACT(DAY FROM v.tb010_012_data) AS dia,
        EXTRACT(MONTH FROM v.tb010_012_data) AS mes,
        EXTRACT(YEAR FROM v.tb010_012_data) AS ano,
        v.tb012_cod_produto AS produto,
        p.tb013_cod_categoria AS categoria,
        p.tipo_produto AS tipo,
        v.tb010_cpf AS cliente
    FROM
        staging.vendas v,
        staging.produtos p
    WHERE
        v.tb012_cod_produto = p.tb012_cod_produto
);

// considerando categoria, tipo e quantidade
INSERT INTO
dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo,
cliente)
(
    SELECT
        SUM(v.tb010_012_quantidade) AS quantidade,
        NULL AS valor,
        NULL AS dia,
        NULL AS mes,
        NULL AS ano,
        NULL AS produto,
        p.tb013_cod_categoria AS categoria,
        p.tipo_produto AS tipo,
        NULL AS cliente
    FROM
```



```

        staging.vendas v,
        staging.produtos p
WHERE
    v.tb012_cod_produto = p.tb012_cod_produto
GROUP BY
    p.tb013_cod_categoria,
    p.tipo_produto
);

// consideranco quantidade, valor, dia, mes, ano e cliente
INSERT INTO
dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo,
cliente)
(
    SELECT
        SUM(v.tb010_012_quantidade) AS quantidade,
        SUM(v.tb010_012_quantidade*v.tb010_012_valor_unitario) AS valor,
        EXTRACT(DAY FROM v.tb010_012_data) AS dia,
        EXTRACT(MONTH FROM v.tb010_012_data) AS mes,
        EXTRACT(YEAR FROM v.tb010_012_data) AS ano,
        NULL AS produto,
        NULL AS categoria,
        NULL AS tipo,
        v.tb010_cpf AS cliente
    FROM
        staging.vendas v
    GROUP BY
        EXTRACT(DAY FROM v.tb010_012_data),
        EXTRACT(MONTH FROM v.tb010_012_data),
        EXTRACT(YEAR FROM v.tb010_012_data),
        v.tb010_cpf
);

// consideranco quantidade, valor, mes, ano e cliente
INSERT INTO
dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo,
cliente)
(
    SELECT
        SUM(v.tb010_012_quantidade) AS quantidade,
        SUM(v.tb010_012_quantidade*v.tb010_012_valor_unitario) AS valor,
        NULL AS dia,
        EXTRACT(MONTH FROM v.tb010_012_data) AS mes,

```

```

        EXTRACT(YEAR FROM v.tb010_012_data) AS ano,
        NULL AS produto,
        NULL AS categoria,
        NULL AS tipo,
        v.tb010_cpf AS cliente
    FROM
        staging.vendas v
    GROUP BY
        EXTRACT(MONTH FROM v.tb010_012_data),
        EXTRACT(YEAR FROM v.tb010_012_data),
        v.tb010_cpf
);

// considerando quantidade, valor, ano e cliente
INSERT INTO
dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo,
cliente)
(
    SELECT
        SUM(v.tb010_012_quantidade) AS quantidade,
        SUM(v.tb010_012_quantidade*v.tb010_012_valor_unitario) AS valor,
        NULL AS dia,
        NULL AS mes,
        EXTRACT(YEAR FROM v.tb010_012_data) AS ano,
        NULL AS produto,
        NULL AS categoria,
        NULL AS tipo,
        v.tb010_cpf AS cliente
    FROM
        staging.vendas v
    GROUP BY
        EXTRACT(YEAR FROM v.tb010_012_data),
        v.tb010_cpf
);

// considerando quantidade, valor e cliente
INSERT INTO
dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo,
cliente)
(
    SELECT
        SUM(v.tb010_012_quantidade) AS quantidade,
        SUM(v.tb010_012_quantidade*v.tb010_012_valor_unitario) AS valor,

```

```

    NULL AS dia,
    NULL AS mes,
    NULL AS ano,
    NULL AS produto,
    NULL AS categoria,
    NULL AS tipo,
    v.tb010_cpf AS cliente
FROM
    staging.vendas v
GROUP BY
    v.tb010_cpf
);

```

5.5.3. Inserção de Dados na Tabela Fato Lucro

```

// Inserção de dados na tabela fato Lucro
INSERT INTO dw.lucro (produto, quantidade, lucro)
(
    SELECT
        v.tb012_cod_produto AS produto,
        SUM(v.tb010_012_quantidade) AS quantidade,
        SUM(v.tb010_012_quantidade * (p.preco_medio_venda -
            p.preco_medio_compra)) AS lucro
    FROM
        staging.vendas v,
        staging.produtos p
    WHERE
        v.tb012_cod_produto = p.tb012_cod_produto AND
        p.preco_medio_compra IS NOT NULL
    GROUP BY
        v.tb012_cod_produto
);

// lucro total
INSERT INTO dw.lucro (produto, quantidade, lucro)
(
    SELECT
        NULL AS produto,
        NULL AS quantidade,
        SUM(v.tb010_012_quantidade * (p.preco_medio_venda -
            p.preco_medio_compra)) AS lucro
    FROM
        staging.vendas v,

```

```

    staging.produtos p
WHERE
    v.tb012_cod_produto = p.tb012_cod_produto AND
    p.preco_medio_compra IS NOT NULL
);

```

6. ANÁLISE GERENCIAL

Analisando as necessidades da empresa foram mapeadas as seguintes necessidades que devem ser respondidas pelo banco de dados multidimensional a ser implementado.

6.1. Quantidade de produtos vendidos por tipo e categoria ao longo do tempo

```

SELECT
    t.descricao AS tipo,
    c.descricao AS categoria,
    SUM(v.quantidade) AS quantidade
FROM
    dw.venda v,
    dw.tipo t,
    dw.categoria c
WHERE
    v.categoria = c.id AND
    v.tipo = t.id AND
    v.quantidade IS NOT NULL AND
    v.valor IS NULL AND
    v.dia IS NULL AND
    v.mes IS NULL AND
    v.ano IS NULL AND
    v.produto IS NULL AND
    v.categoria IS NOT NULL AND
    v.tipo IS NOT NULL AND
    v.cliente IS NULL
GROUP BY
    t.descricao,
    c.descricao;

```

6.1.1. Evidência Análise Gerencial

SELECT t.descricao AS tipo, c.descricao AS categoria, SUM(v.qua

Grade	ABC tipo	ABC categoria	123 quantidade
1	alimento	Alimentos Perecíveis	21
2	eletro	CD e DVD	6
3	eletro	Eletrodomésticos	9
4	vestuario	Roupas Femininas	8
5	vestuario	Roupas Infantis	7
6	vestuario	Roupas Masculinas	4

6.2. Quantidade e valor total das compras por produto, possibilitando visão hierárquica ao longo do tempo

// Visão considerando produto, ano e mês

```
SELECT
  c.produto,
  p.descricao,
  SUM(c.quantidade) AS quantidade,
  SUM(c.valor) AS valor,
  c.ano,
  c.mes
FROM
  dw.compra c,
  dw.produto p
WHERE
  p.id = c.produto AND
  c.mes IS NOT NULL AND -- * c.mes = 4
  c.ano IS NOT NULL -- * c.ano = 2010
GROUP BY
  c.produto,
  p.descricao,
  c.ano,
  c.mes;
```

// Visão considerando produto e mês

```
SELECT
  c.produto,
  p.descricao,
  SUM(c.quantidade) AS quantidade,
  SUM(c.valor) AS valor,
```

```

        c.mes
FROM
    dw.compra c,
    dw.produto p
WHERE
    p.id = c.produto AND
    c.mes IS NOT NULL AND -- * c.mes = 4
    c.ano IS NULL
GROUP BY
    c.produto,
    p.descricao,
    c.mes;

// Visão considerando produto e ano
SELECT
    c.produto,
    p.descricao,
    SUM(c.quantidade) AS quantidade,
    SUM(c.valor) AS valor,
    c.ano
FROM
    dw.compra c,
    dw.produto p
WHERE
    p.id = c.produto AND
    c.ano IS NOT NULL AND -- * c.ano = 2010
    c.mes IS NULL
GROUP BY
    c.produto,
    p.descricao,
    c.ano;

// Visão considerando mês e ano
SELECT
    SUM(c.quantidade) AS quantidade,
    SUM(c.valor) AS valor,
    c.mes,
    c.ano
FROM
    dw.compra c
WHERE

```



```

    c.produto IS NULL AND
    c.mes IS NOT NULL AND -- * c.mes = 4
    c.ano IS NOT NULL -- * c.ano = 2010
GROUP BY
    c.mes,
    c.ano;

// Visão considerando apenas mês
SELECT
    SUM(c.quantidade) AS quantidade,
    SUM(c.valor) AS valor,
    c.mes
FROM
    dw.compra c
WHERE
    c.produto IS NULL AND
    c.mes IS NOT NULL AND -- * c.mes = 4
    c.ano IS NULL
GROUP BY
    c.mes;

// Visão considerando apenas ano
SELECT
    SUM(c.quantidade) AS quantidade,
    SUM(c.valor) AS valor,
    c.ano
FROM
    dw.compra c
WHERE
    c.produto IS NULL AND
    c.mes IS NULL AND
    c.ano IS NOT NULL -- * c.ano = 2010
GROUP BY
    c.ano;

// Visão considerando apenas produto
SELECT
    c.produto,
    p.descricao,
    SUM(c.quantidade) AS quantidade,

```

```

        SUM(c.valor) AS valor
FROM
    dw.compra c,
    dw.produto p
WHERE
    p.id = c.produto AND
    c.ano IS NULL AND
    c.mes IS NULL
GROUP BY
    c.produto,
    p.descricao;

// Visão sem considerar nada
SELECT
    SUM(c.quantidade) AS quantidade,
    SUM(c.valor) AS valor
FROM
    dw.compra c
WHERE
    c.produto IS NULL AND
    c.mes IS NULL AND
    c.ano IS NULL;

```

6.2.1. Evidências Análise Gerencial

6.2.1.1. Visão considerando produto, ano e mês

SELECT c.produto, p.descricao, SUM(c.quantidade) AS quantidade Enter a SQL expression to filter results (use Ctrl+Space							
Grade	123 produto	ABC descricao	123 quantidade	123 valor	123 ano	123 mes	
1	71	Calça Moletom	6	178,5	2.010	10	
2	80	Camiseta	7	118,93	2.010	4	
3	57	DVD Coletânea	6	64,92	2.010	8	
4	71	Calça Moletom	7	208,25	2.010	9	
5	70	Calça Jeans	6	611,94	2.010	8	
6	10	Biscoito Recheado	1	0,59	2.010	2	
7	12	logurte	3	2,07	2.010	8	
8	53	CD Caipira	4	43,2	2.010	4	
9	13	Barra de Chocolate	3	6,63	2.010	11	
10	11	Pão-de-queijo Congelado	3	4,62	2.010	4	
11	30	Geladeira	5	3.601,95	2.010	10	
12	11	Pão-de-queijo Congelado	2	4,02	2.010	6	
13	59	DVD Virgem	5	2,15	2.010	10	
14	70	Calça Jeans	8	679,92	2.010	7	
15	13	Barra de Chocolate	4	8,36	2.010	12	
16	10	Biscoito Recheado	3	2,67	2.010	1	
17	81	Bermuda	8	135,92	2.010	6	
18	56	DVD POP	2	23,24	2.010	7	
19	10	Biscoito Recheado	2	1,38	2.010	3	
20	11	Pão-de-queijo Congelado	1	1,64	2.010	5	
21	82	Tênis	5	169,95	2.010	8	
22	58	DVD Caipira	6	69,78	2.010	9	
23	12	logurte	3	2,07	2.010	7	
24	82	Tênis	8	407,92	2.010	9	
25	30	Geladeira	5	3.554,95	2.010	11	
26	79	Bolsas	3	303,45	2.010	3	
27	68	Roupas de Baixo	6	81,18	2.010	5	
28	12	logurte	3	1,56	2.010	9	
29	80	Camiseta	7	154,63	2.010	5	
30	68	Roupas de Baixo	6	81,42	2.010	4	
31	55	DVD Rock	3	32,43	2.010	6	
32	83	Bonés	8	71,92	2.010	10	
33	54	CD Virgem	6	69,66	2.010	5	
34	69	Gravatas	9	152,91	2.010	6	
35	81	Bermuda	4	88,36	2.010	7	
36	67	Meias	9	92,34	2.010	3	
37	52	CD Coletânea	6	69,6	2.010	3	
38	13	Barra de Chocolate	3	7,05	2.010	10	

6.2.1.2. Visão considerando produto e mês

SELECT c.produto, p.descricao, SUM(c.quantidade) AS quantidade						Enter a SQL expression to filter results	
Grade	123 produto	ABC descricao	123 quantidade	123 valor	123 mes		
1	70	Calça Jeans	6	611,94	8		
2	83	Bonés	8	71,92	10		
3	80	Camiseta	7	118,93	4		
4	11	Pão-de-queijo Congelado	2	4,02	6		
5	12	Iogurte	3	1,56	9		
6	13	Barra de Chocolate	4	8,36	12		
7	10	Biscoito Recheado	1	0,59	2		
8	68	Roupas de Baixo	6	81,42	4		
9	82	Tênis	5	169,95	8		
10	13	Barra de Chocolate	3	7,05	10		
11	54	CD Virgem	6	69,66	5		
12	12	Iogurte	3	2,07	8		
13	12	Iogurte	3	2,07	7		
14	30	Geladeira	5	3.601,95	10		
15	71	Calça Moleton	7	208,25	9		
16	52	CD Coletânea	6	69,6	3		
17	79	Bolsas	3	303,45	3		
18	56	DVD POP	2	23,24	7		
19	81	Bermuda	4	88,36	7		
20	69	Gravatas	9	152,91	6		
21	68	Roupas de Baixo	6	81,18	5		
22	59	DVD Virgem	5	2,15	10		
23	10	Biscoito Recheado	2	1,38	3		
24	53	CD Caipira	4	43,2	4		
25	30	Geladeira	5	3.554,95	11		
26	57	DVD Coletânea	6	64,92	8		
27	13	Barra de Chocolate	3	6,63	11		
28	71	Calça Moleton	6	178,5	10		
29	55	DVD Rock	3	32,43	6		
30	82	Tênis	8	407,92	9		
31	58	DVD Caipira	6	69,78	9		
32	81	Bermuda	8	135,92	6		
33	67	Meias	9	92,34	3		
34	10	Biscoito Recheado	3	2,67	1		
35	70	Calça Jeans	8	679,92	7		
36	11	Pão-de-queijo Congelado	3	4,62	4		
37	11	Pão-de-queijo Congelado	1	1,64	5		
38	80	Camiseta	7	154,63	5		

6.2.1.3. Visão considerando produto e ano

SELECT c.produto, p.descricao, SUM(c.quantidade) AS quantidade | Enter a SQL expression to filter result

Grade	123 produto	ABC descricao	123 quantidade	123 valor	123 ano
1	59	DVD Virgem	5	2,15	2.010
2	10	Biscoito Recheado	6	4,64	2.010
3	81	Bermuda	12	224,28	2.010
4	55	DVD Rock	3	32,43	2.010
5	68	Roupas de Baixo	12	162,6	2.010
6	52	CD Coletânea	6	69,6	2.010
7	11	Pão-de-queijo Congelado	6	10,28	2.010
8	30	Geladeira	10	7.156,9	2.010
9	12	Iogurte	9	5,7	2.010
10	83	Bonés	8	71,92	2.010
11	80	Camiseta	14	273,56	2.010
12	56	DVD POP	2	23,24	2.010
13	58	DVD Caipira	6	69,78	2.010
14	67	Meias	9	92,34	2.010
15	82	Tênis	13	577,87	2.010
16	57	DVD Coletânea	6	64,92	2.010
17	70	Calça Jeans	14	1.291,86	2.010
18	79	Bolsas	3	303,45	2.010
19	13	Barra de Chocolate	10	22,04	2.010
20	54	CD Virgem	6	69,66	2.010
21	71	Calça Moletom	13	386,75	2.010
22	53	CD Caipira	4	43,2	2.010
23	69	Gravatas	9	152,91	2.010

6.2.1.4. Visão considerando mês e ano

SELECT SUM(c.quantidade) AS quantidade, SUM(c.valor) AS valc | Enter

Grade	123 quantidade	123 valor	123 mes	123 ano
1	3	2,67	1	2.010
2	20	848,88	8	2.010
3	20	248,17	4	2.010
4	8	3.561,58	11	2.010
5	22	325,28	6	2.010
6	20	307,11	5	2.010
7	24	687,51	9	2.010
8	17	793,59	7	2.010
9	27	3.861,57	10	2.010
10	1	0,59	2	2.010
11	20	466,77	3	2.010
12	4	8,36	12	2.010

6.2.1.5. Visão considerando apenas mês

SELECT SUM(c.quantidade) AS quantidade, SUM(c.valor) AS valor

Grade	123 quantidade	123 valor	123 mes
1	4	8,36	12
2	20	466,77	3
3	8	3.561,58	11
4	20	848,88	8
5	17	793,59	7
6	27	3.861,57	10
7	24	687,51	9
8	3	2,67	1
9	20	307,11	5
10	1	0,59	2
11	20	248,17	4
12	22	325,28	6

6.2.1.6. Visão considerando apenas ano

SELECT SUM(c.quantidade) AS quantidade, SUM(c.valor) AS valor

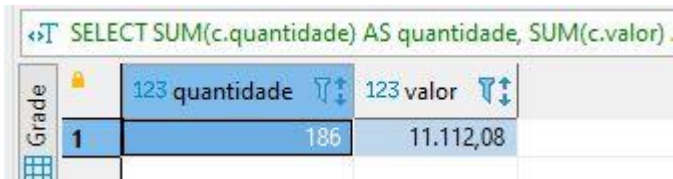
Grade	123 quantidade	123 valor	123 ano
1	186	11.112,08	2.010

6.2.1.7. Visão considerando apenas produto

SELECT c.produto, p.descricao, SUM(c.quantidade) AS quantidade, SUM(c.valor) AS valor

Grade	123 produto	ABC descricao	123 quantidade	123 valor
1	55	DVD Rock	3	32,43
2	52	CD Coletânea	6	69,6
3	58	DVD Caipira	6	69,78
4	53	CD Caipira	4	43,2
5	59	DVD Virgem	5	2,15
6	67	Meias	9	92,34
7	68	Roupas de Baixo	12	162,6
8	69	Gravatas	9	152,91
9	79	Bolsas	3	303,45
10	80	Camiseta	14	273,56
11	70	Calça Jeans	14	1.291,86
12	12	Iogurte	9	5,7
13	82	Tênis	13	577,87
14	10	Biscoito Recheado	6	4,64
15	83	Bonés	8	71,92

6.2.1.8. Visão sem considerar nada



The screenshot shows a SQL query editor with a query window at the top and a results grid below. The query is: `SELECT SUM(c.quantidade) AS quantidade, SUM(c.valor)`. The results grid has a 'Grade' column on the left and two columns for the query results. The first row shows '123' for both 'quantidade' and 'valor'. The second row shows '186' for 'quantidade' and '11.112,08' for 'valor'.

Grade	quantidade	valor
	123	123
1	186	11.112,08

6.3. Clientes que mais gastaram na loja virtual com quantidade acumulada, valor acumulado e média em um determinado período

```
// Período considerado dia, mes e ano
SELECT
  v.cliente as cliente,
  SUM(v.quantidade) AS quantidade_acumulada,
  SUM(v.valor) AS valor_acumulado,
  AVG(v.valor) AS valor_médio
FROM
  dw.venda v
WHERE
  v.dia BETWEEN 1 AND 31 AND
  v.mes BETWEEN 3 AND 5 AND
  v.ano BETWEEN 2009 AND 2011 AND
  v.quantidade IS NOT NULL AND
  v.valor IS NOT NULL AND
  v.dia IS NOT NULL AND
  v.mes IS NOT NULL AND
  v.ano IS NOT NULL AND
  v.produto IS NULL AND
  v.categoria IS NULL AND
  v.tipo IS NULL AND
  v.cliente IS NOT NULL
GROUP BY
  v.cliente
ORDER BY
  SUM(v.valor) DESC;

// Período considerando mes e ano
SELECT
  v.cliente as cliente,
  SUM(v.quantidade) AS quantidade_acumulada,
  SUM(v.valor) AS valor_acumulado,
  AVG(v.valor) AS valor_médio
FROM
```

```

    dw.venda v
WHERE
    v.mes BETWEEN 3 AND 5 AND
    v.ano BETWEEN 2009 AND 2011 AND
    v.quantidade IS NOT NULL AND
    v.valor IS NOT NULL AND
    v.dia IS NULL AND
    v.mes IS NOT NULL AND
    v.ano IS NOT NULL AND
    v.produto IS NULL AND
    v.categoria IS NULL AND
    v.tipo IS NULL AND
    v.cliente IS NOT NULL
GROUP BY
    v.cliente
ORDER BY
    SUM(v.valor) DESC;

// Período considerando apenas ano
SELECT
    v.cliente as cliente,
    SUM(v.quantidade) AS quantidade_acumulada,
    SUM(v.valor) AS valor_acumulado,
    AVG(v.valor) AS valor_médio
FROM
    dw.venda v
WHERE
    v.ano BETWEEN 2009 AND 2011 AND
    v.quantidade IS NOT NULL AND
    v.valor IS NOT NULL AND
    v.dia IS NULL AND
    v.mes IS NULL AND
    v.ano IS NOT NULL AND
    v.produto IS NULL AND
    v.categoria IS NULL AND
    v.tipo IS NULL AND
    v.cliente IS NOT NULL
GROUP BY
    v.cliente
ORDER BY
    SUM(v.valor) DESC;

// Período total

```

```

SELECT
    v.cliente as cliente,
    SUM(v.quantidade) AS quantidade_acumulada,
    SUM(v.valor) AS valor_acumulado,
    AVG(v.valor) AS valor_médio
FROM
    dw.venda v
WHERE
    v.quantidade IS NOT NULL AND
    v.valor IS NOT NULL AND
    v.dia IS NULL AND
    v.mes IS NULL AND
    v.ano IS NOT NULL AND
    v.produto IS NULL AND
    v.categoria IS NULL AND
    v.tipo IS NULL AND
    v.cliente IS NOT NULL
GROUP BY
    v.cliente
ORDER BY
    SUM(v.valor) DESC;

```

6.3.1. Evidências Análise Gerencial

6.3.1.1. Período considerando dia, mês e ano

	123 cliente	123 quantidade_acumulada	123 valor_acumulado	123 valor_médio
1	10.000.000.028	5	3.648,22	1.824,11
2	10.000.000.029	3	2.397,54	1.198,77
3	10.000.000.007	3	407,94	407,94
4	10.000.000.008	1	163,18	163,18
5	10.000.000.015	2	108,76	108,76
6	10.000.000.013	2	108,76	54,38
7	10.000.000.010	2	95,2	95,2
8	10.000.000.011	1	50,32	50,32
9	10.000.000.014	2	49,72	24,86
10	10.000.000.009	1	47,6	47,6
11	10.000.000.024	5	41,92	20,96
12	10.000.000.012	1	35,34	35,34
13	10.000.000.006	1	27,18	27,18
14	10.000.000.026	3	20,81	10,405
15	10.000.000.025	2	18,94	9,47
16	10.000.000.027	6	10,26	5,13
17	10.000.000.022	3	3,3	3,3
18	10.000.000.023	2	1,66	1,66
19	10.000.000.021	1	1,1	1,1

6.3.1.2. Período considerando mês e ano

SELECT v.cliente as cliente, SUM(v.quantidade) AS quantidade_a

Grade	123 cliente	123 quantidade_acumulada	123 valor_acumulado	123 valor_médio
1	10.000.000.028	5	3.648,22	3.648,22
2	10.000.000.029	3	2.397,54	2.397,54
3	10.000.000.007	3	407,94	407,94
4	10.000.000.008	1	163,18	163,18
5	10.000.000.013	2	108,76	108,76
6	10.000.000.015	2	108,76	108,76
7	10.000.000.010	2	95,2	95,2
8	10.000.000.011	1	50,32	50,32
9	10.000.000.014	2	49,72	49,72
10	10.000.000.009	1	47,6	47,6
11	10.000.000.024	5	41,92	41,92
12	10.000.000.012	1	35,34	35,34
13	10.000.000.006	1	27,18	27,18
14	10.000.000.026	3	20,81	20,81
15	10.000.000.025	2	18,94	18,94
16	10.000.000.027	6	10,26	10,26
17	10.000.000.022	3	3,3	3,3
18	10.000.000.023	2	1,66	1,66
19	10.000.000.021	1	1,1	1,1

6.3.1.3. Período considerando apenas ano

SELECT v.cliente as cliente, SUM(v.quantidade) AS quantidade_a

Grade	123 cliente	123 quantidade_acumulada	123 valor_acumulado	123 valor_médio
1	10.000.000.028	5	3.648,22	3.648,22
2	10.000.000.030	2	3.520	3.520
3	10.000.000.029	3	2.397,54	2.397,54
4	10.000.000.031	1	1.600	1.600
5	10.000.000.032	1	574,4	574,4
6	10.000.000.007	3	407,94	407,94
7	10.000.000.008	1	163,18	163,18
8	10.000.000.013	2	108,76	108,76
9	10.000.000.015	2	108,76	108,76
10	10.000.000.010	2	95,2	95,2
11	10.000.000.011	1	50,32	50,32
12	10.000.000.014	2	49,72	49,72
13	10.000.000.009	1	47,6	47,6
14	10.000.000.024	5	41,92	41,92
15	10.000.000.012	1	35,34	35,34
16	10.000.000.006	1	27,18	27,18
17	10.000.000.026	3	20,81	20,81
18	10.000.000.025	2	18,94	18,94
19	10.000.000.027	6	10,26	10,26

6.3.1.4. Período total

SELECT v.cliente as cliente, SUM(v.quantidade) AS quantidade_a

Grade	123 cliente	123 quantidade_acumulada	123 valor_acumulado	123 valor_médio
1	10.000.000.028	5	3.648,22	3.648,22
2	10.000.000.030	2	3.520	3.520
3	10.000.000.029	3	2.397,54	2.397,54
4	10.000.000.031	1	1.600	1.600
5	10.000.000.032	1	574,4	574,4
6	10.000.000.007	3	407,94	407,94
7	10.000.000.008	1	163,18	163,18
8	10.000.000.013	2	108,76	108,76
9	10.000.000.015	2	108,76	108,76
10	10.000.000.010	2	95,2	95,2
11	10.000.000.011	1	50,32	50,32
12	10.000.000.014	2	49,72	49,72
13	10.000.000.009	1	47,6	47,6
14	10.000.000.024	5	41,92	41,92
15	10.000.000.012	1	35,34	35,34
16	10.000.000.006	1	27,18	27,18
17	10.000.000.026	3	20,81	20,81
18	10.000.000.025	2	18,94	18,94
19	10.000.000.027	6	10,26	10,26
20	10.000.000.022	3	3,3	3,3
21	10.000.000.000	2	2,84	2,84
22	10.000.000.001	3	2,82	2,82
23	10.000.000.023	2	1,66	1,66
24	10.000.000.021	1	1,1	1,1

6.4. Últimas compras realizadas por cliente, e tempo decorrido até a data atual

```
SELECT
    v.cliente AS cliente,
    c.nome AS nome,
    make_date(v.ano, v.mes, v.dia) AS data_compra,
    AGE(NOW():: date, make_date(v.ano, v.mes, v.dia)) AS
tempo_decorrido
FROM
    dw.venda v,
    dw.cliente c
WHERE
    v.cliente = c.cpf AND
    v.quantidade IS NOT NULL AND
    v.valor IS NOT NULL AND
```



```

v.dia IS NOT NULL AND
v.mes IS NOT NULL AND
v.ano IS NOT NULL AND
v.produto IS NOT NULL AND
v.categoria IS NOT NULL AND
v.tipo IS NOT NULL AND
v.cliente IS NOT NULL
ORDER BY
v.ano DESC,
v.mes DESC,
v.dia DESC

```

6.4.1. Evidência Análise Gerencial

SELECT v.cliente AS cliente, c.nome AS nome, make_date(v.ano, Enter a SQL expression to				
Grade	123 cliente	ABC nome	data_compra	tempo_decorrido
1	10.000.000.032	NOME Teste 33	2010-12-04	11 years 7 mons 28 days
2	10.000.000.001	NOME Teste 02	2010-12-03	11 years 7 mons 29 days
3	10.000.000.031	NOME Teste 32	2010-11-04	11 years 8 mons 27 days
4	10.000.000.000	NOME Teste 01	2010-11-03	11 years 8 mons 28 days
5	10.000.000.030	NOME Teste 31	2010-10-04	11 years 9 mons 28 days
6	10.000.000.029	NOME Teste 30	2010-04-09	12 years 3 mons 22 days
7	10.000.000.028	NOME Teste 29	2010-04-08	12 years 3 mons 23 days
8	10.000.000.027	NOME Teste 28	2010-04-07	12 years 3 mons 24 days
9	10.000.000.029	NOME Teste 30	2010-04-06	12 years 3 mons 25 days
10	10.000.000.014	NOME Teste 15	2010-04-06	12 years 3 mons 25 days
11	10.000.000.011	NOME Teste 12	2010-04-06	12 years 3 mons 25 days
12	10.000.000.026	NOME Teste 27	2010-04-06	12 years 3 mons 25 days
13	10.000.000.013	NOME Teste 14	2010-04-05	12 years 3 mons 26 days
14	10.000.000.025	NOME Teste 26	2010-04-05	12 years 3 mons 26 days
15	10.000.000.010	NOME Teste 11	2010-04-05	12 years 3 mons 26 days
16	10.000.000.028	NOME Teste 29	2010-04-05	12 years 3 mons 26 days
17	10.000.000.024	NOME Teste 25	2010-04-04	12 years 3 mons 27 days
18	10.000.000.015	NOME Teste 16	2010-04-04	12 years 3 mons 27 days
19	10.000.000.027	NOME Teste 28	2010-04-04	12 years 3 mons 27 days
20	10.000.000.009	NOME Teste 10	2010-04-04	12 years 3 mons 27 days
21	10.000.000.026	NOME Teste 27	2010-04-03	12 years 3 mons 28 days
22	10.000.000.014	NOME Teste 15	2010-04-03	12 years 3 mons 28 days
23	10.000.000.023	NOME Teste 24	2010-04-03	12 years 3 mons 28 days
24	10.000.000.008	NOME Teste 09	2010-04-03	12 years 3 mons 28 days
25	10.000.000.013	NOME Teste 14	2010-04-02	12 years 3 mons 29 days
26	10.000.000.025	NOME Teste 26	2010-04-02	12 years 3 mons 29 days
27	10.000.000.022	NOME Teste 23	2010-04-02	12 years 3 mons 29 days
28	10.000.000.007	NOME Teste 08	2010-04-02	12 years 3 mons 29 days

6.5. Lucratividade bruta dos produtos comprados e posteriormente vendidos

```
// lucro por produto
SELECT
  p.descricao AS produto,
  l.quantidade AS quantidade_vendida,
  l.lucro AS lucro_total
FROM
  dw.lucro l,
  dw.produto p
WHERE
  l.produto = p.id AND
  l.produto IS NOT NULL AND
  l.quantidade IS NOT NULL AND
  l.lucro IS NOT NULL

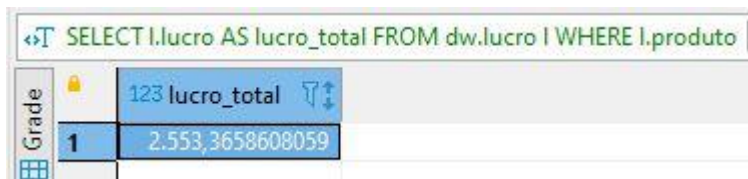
// lucro total
SELECT
  l.lucro AS lucro_total
FROM
  dw.lucro l
WHERE
  l.produto IS NULL AND
  l.quantidade IS NULL AND
  l.lucro IS NOT NULL
```

6.5.1. Evidências Análise Gerencial

6.5.1.1. Lucro por produto

SELECT p.descricao AS produto, l.quantidade AS quantidade_ver Enter a SQL			
Grade	ABC produto	123 quantidade_vendida	123 lucro_total
1	Calça Moletom	3	53,55
2	Camiseta	1	15,8
3	Calça Jeans	4	202,0171428571
4	Bonés	1	5,39
5	Biscoito Recheado	5	1,7933333333
6	Gravatas	1	10,19
7	DVD Virgem	2	0,52
8	Tênis	3	56,9853846154
9	DVD Coletânea	1	6,49
10	Bermuda	2	25,14
11	Geladeira	5	2.154,57
12	DVD POP	2	13,94
13	DVD Caipira	1	6,98

6.5.1.2. Lucro total



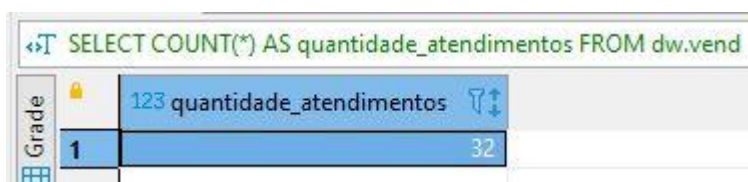
The screenshot shows a SQL query in a text editor: `SELECT l.lucro AS lucro_total FROM dw.lucro l WHERE l.produto`. Below the query, a table with two columns is displayed. The first column is labeled 'Grade' and contains the value '1'. The second column is labeled '123 lucro_total' and contains the value '2.553,3658608059'.

Grade	123 lucro_total
1	2.553,3658608059

6.6. Quantidade de atendimentos realizados ao longo do tempo.

```
SELECT
    COUNT(*) AS quantidade_atendimentos
FROM
    dw.venda v
WHERE
    v.quantidade IS NOT NULL AND
    v.valor IS NOT NULL AND
    v.dia IS NOT NULL AND
    v.mes IS NOT NULL AND
    v.ano IS NOT NULL AND
    v.produto IS NOT NULL AND
    v.categoria IS NOT NULL AND
    v.tipo IS NOT NULL AND
    v.cliente IS NOT NULL;
```

6.6.1. Evidência Análise Gerencial



The screenshot shows a SQL query in a text editor: `SELECT COUNT(*) AS quantidade_atendimentos FROM dw.vend`. Below the query, a table with two columns is displayed. The first column is labeled 'Grade' and contains the value '1'. The second column is labeled '123 quantidade_atendimentos' and contains the value '32'.

Grade	123 quantidade_atendimentos
1	32

7. FREQUÊNCIA DE CARGA

A frequência estabelecida das execuções ocorrerão diariamente, preferencialmente fora do expediente de trabalho a fim de não prejudicar o estabelecimento. A periodicidade foi estabelecida para que a loja possua um controle das vendas. O primeiro levantamento de dados será completo, e demandará mais tempo, já as execuções posteriores apenas incrementarão as cargas e serão mais rápidas.

8. TEMPO DE RETENÇÃO

O tempo de retenção do banco de dados da Staging Area foi definido como no máximo uma semana, uma vez que será alimentado também semanalmente.

O tempo de retenção do Data Warehouse terá um tempo de 5 anos, onde a retenção de dados será controlada através de backups diários. Os dados serão distribuídos entre as lojas existentes levando em conta as informações necessárias para cada uma delas, e a sede principal será responsável por armazenar os dados de todas unidades.

9. HARDWARE E SGBDS RECOMENDADOS

Para execução da Análises Gerenciais, recomendamos o SGBD PostgreSQL, uma vez que o próprio DW já foi desenvolvido utilizando essa ferramenta, que apresenta um ótimo desempenho, e é gratuita para uso.

Quanto ao Hardware, a equipe indica a utilização de um servidor dedicado com a seguinte configuração:

- Processador: AMD Epyc 7742 2.25GHz, 64C/128T, 256M Cache.
- Placa Mãe: Supermicro H11SSL-NC Rev 2
- Memória RAM: 128Gb DDR4 ECC
- Armazenamento: SSD NAS 500Gb para o sistema operacional e HDD NAS 16TB
- 2 Fontes 750W com redundância

10. CONSIDERAÇÕES FINAIS

Usar um data warehouse para extrair informações de gerenciamento pode não parecer fundamental para uma empresa inicialmente, mas à medida que a esta tem um crescimento, a extração dessas informações diretamente do servidor e de banco de dados operacionais pode ser custoso do ponto de vista de desempenho, pode atrapalhar as tarefas e rotinas do dia-a-dia, podendo até afetar custos, já que essas análises influenciam em tomadas de decisão que podem prejudicar a organização.

As soluções de data warehouse apresentadas aqui demonstram que sua implementação pode ser simples, podendo utilizar inclusive ferramentas gratuitas. Ou pode-se optar pela utilização de ferramentas mais otimizadas e poderosas, levando em consideração que isso deve ser visto como investimento, facilitando o acesso a informações que auxiliam em tomadas de decisão assertivas no que se refere aos objetivos da empresa.

11. REFERÊNCIAS

<<https://forums.pentaho.com/threads/217203-Error-while-connecting-to-MSSQL-DB/>>

<<https://sourceforge.net/projects/jtds/>>

<<https://www.tecmint.com/convert-files-to-utf-8-encoding-in-linux/>>

<<https://conteudo.precocerto.co/margem-bruta-margem-liquida/#:~:text=A%20margem%20de%20lucro%20bruta,faturamento%20que%20%C3%A9%20lucro%20%C3%ADquido.>>

12. INFORMAÇÕES COMPLEMENTARES

12.1. Ferramentas Utilizadas

- Microsoft SQL Server 2019
- Microsoft SQL Server Management Studio
- DBeaver
- Pentaho
- Driver MSSQL Server para o Pentaho:
 - <<https://sourceforge.net/projects/jtds/files/latest/download>>

12.2. Evidências

12.2.1. Restauração do Arquivo de Backup no MS SQL Server

12.2.1.1. Importação Dados no MS SQL Server

```
--USE master
--GO
--DROP DATABASE ADS
--GO

--CREATE DATABASE ADS
--GO

USE ADS
GO

DROP TABLE tb010_clientes_antigos
go

DROP TABLE tb016_prd_vestuarios
go

DROP TABLE tb011_logins
go

DROP TABLE tb015_prd_eletros
go

DROP TABLE tb014_prd_alimentos
go

DROP TABLE tb005_006_funcionarios_cargos
go

DROP TABLE tb006_cargos
go

DROP TABLE tb010_012_vendas
go

DROP TABLE tb010_clientes
```

12.2.1.2. Evidência da Execução do Script para Restaurar o Backup

```
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 55, 5, '03/06/2010', 3, 10.81)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 56, 6, '01/07/2010', 2, 11.62)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 57, 8, '05/08/2010', 6, 10.82)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 58, 7, '05/09/2010', 6, 11.63)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 59, 7, '05/10/2010', 5, 0.43)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 60, 9, '05/24/2010', 4, 59.49)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 60, 10, '05/25/2010', 6, 76.49)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 61, 11, '05/26/2010', 5, 18.71)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 61, 13, '05/27/2010', 3, 18.73)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 62, 12, '05/28/2010', 5, 18.74)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 62, 13, '05/16/2010', 6, 29.75)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 63, 14, '05/17/2010', 6, 22.95)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 63, 15, '05/18/2010', 6, 32.30)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 64, 16, '05/19/2010', 6, 18.70)
INSERT INTO ADS.dbo.tb012_017_compras VALUES( 64, 17, '05/20/2010', 6, 29.75)
```

100 %

Messages

```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

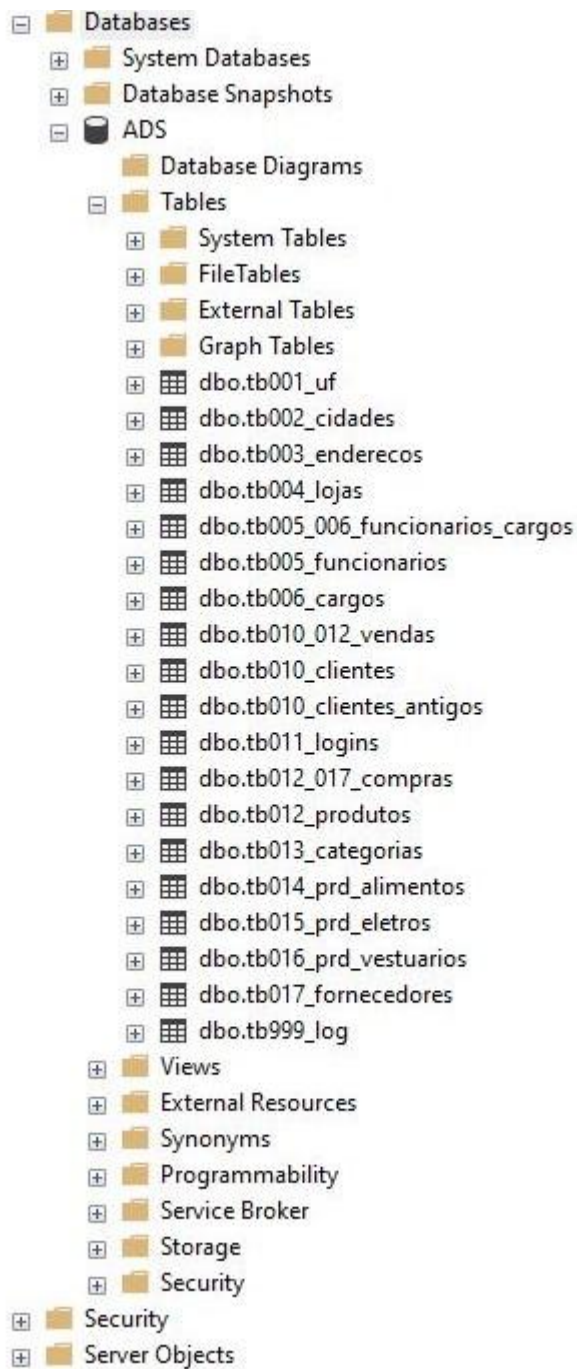
(1 row affected)

(1 row affected)

(0 rows affected)

(1 row affected)
```

12.2.1.3. Evidência da Base e Tabelas Criadas Após Restauração



ANEXOS

-- Adição da coluna tipo produto na tabela produtos de staging area		
ALTER TABLE staging.produtos add tipo_produto int;		
Statistics 1 X		
Name	Value	
Updated Rows	0	
Query	ALTER TABLE staging.produtos add tipo_produto int	

-- Adição da coluna preco medio compra na tabela produtos de staging area		
ALTER TABLE staging.produtos add preco_medio_compra float;		
Statistics 1 X		
Name	Value	
Updated Rows	0	
Query	ALTER TABLE staging.produtos add preco_medio_compra float	

-- Adição da coluna preco medio venda na tabela produtos de staging area		
ALTER TABLE staging.produtos add preco_medio_venda float;		
Statistics 1 X		
Name	Value	
Updated Rows	0	
Query	ALTER TABLE staging.produtos add preco_medio_venda float	

-- Update da coluna tipo_produto na staging area conforme tipo de cada produto		
-- Considerando produtos do tipo alimento		
UPDATE staging.produtos		
SET tipo_produto = 1		
WHERE tb012_cod_produto IN (SELECT p.tb012_cod_produto FROM staging.prđ_alimentos pa, staging.produtos p		
WHERE pa.tb012_cod_produto = p.tb012_cod_produto);		
Statistics 1 X		
Name	Value	
Updated Rows	10	
Query	UPDATE staging.produtos	
	SET tipo_produto = 1	
	WHERE tb012_cod_produto IN (SELECT p.tb012_cod_produto FROM staging.prđ_alimentos pa, staging.produtos p	
	WHERE pa.tb012_cod_produto = p.tb012_cod_produto)	

<pre>-- Considerando produtos do tipo eletro UPDATE staging.produtos SET tipo_produto = 2 WHERE tb012_cod_produto IN (SELECT p.tb012_cod_produto FROM staging.prđ_eletros pa, staging.produtos p WHERE pa.tb012_cod_produto = p.tb012_cod_produto);</pre>		
Statistics 1		
Name	Value	
Updated Rows	30	
Query	<pre>UPDATE staging.produtos SET tipo_produto = 2 WHERE tb012_cod_produto IN (SELECT p.tb012_cod_produto FROM staging.prđ_eletros pa, staging.produtos p WHERE pa.tb012_cod_produto = p.tb012_cod_produto)</pre>	

<pre>-- Considerando produtos do tipo vestuario UPDATE staging.produtos SET tipo_produto = 3 WHERE tb012_cod_produto IN (SELECT p.tb012_cod_produto FROM staging.prđ_vestuarios pa, staging.produtos p WHERE pa.tb012_cod_produto = p.tb012_cod_produto);</pre>		
Statistics 1		
Name	Value	
Updated Rows	24	
Query	<pre>UPDATE staging.produtos SET tipo_produto = 3 WHERE tb012_cod_produto IN (SELECT p.tb012_cod_produto FROM staging.prđ_vestuarios pa, staging.produtos p WHERE pa.tb012_cod_produto = p.tb012_cod_produto)</pre>	

<pre>-- Cálculo da média ponderada do valor pago pelo produto na compra e inserção dos valores na coluna preco_medio_compra da tabela staging.produtos UPDATE staging.produtos p SET preco_medio_compra = subquery.preco_medio_compra FROM (SELECT p.tb012_cod_produto, p.tb012_descricao, sum((c.tb012_017_quantidade*c.tb012_017_valor_unitario))/sum(c.tb012_017_quantidade) AS preco_medio_compra FROM staging.compras c, staging.produtos p GROUP BY p.tb012_cod_produto, c.tb012_cod_produto, p.tb012_descricao HAVING c.tb012_cod_produto = p.tb012_cod_produto) AS subquery WHERE p.tb012_cod_produto = subquery.tb012_cod_produto;</pre>		
Statistics 1		
Name	Value	
Updated Rows	23	
Query	<pre>UPDATE staging.produtos p SET preco_medio_compra = subquery.preco_medio_compra FROM (SELECT p.tb012_cod_produto, p.tb012_descricao, sum((c.tb012_017_quantidade*c.tb012_017_valor_unitario))/sum(c.tb012_017_quantidade) AS preco_medio_compra FROM staging.compras c, staging.produtos p GROUP BY p.tb012_cod_produto, c.tb012_cod_produto, p.tb012_descricao HAVING c.tb012_cod_produto = p.tb012_cod_produto) AS subquery WHERE p.tb012_cod_produto = subquery.tb012_cod_produto</pre>	

-- Cálculo da média ponderada do valor pago pelo produto na venda e inserção dos valores na coluna preco medio venda da tabela staging.produtos

```

UPDATE staging.produtos p
SET preco_medio_venda = subquery.preco_medio_venda
FROM (
  SELECT
    p.tb012_cod_produto,
    p.tb012_descricao,
    sum((v.tb010_012_quantidade * v.tb010_012_valor_unitario))/sum(v.tb010_012_quantidade) AS preco_medio_venda
  FROM
    staging.vendas v,
    staging.produtos p
  GROUP BY
    p.tb012_cod_produto,
    v.tb012_cod_produto,
    p.tb012_descricao
  HAVING
    v.tb012_cod_produto = p.tb012_cod_produto
) AS subquery
WHERE
  p.tb012_cod_produto = subquery.tb012_cod_produto;

```

Name	Value
Updated Rows	20
Query	UPDATE staging.produtos p SET preco_medio_venda = subquery.preco_medio_venda FROM (SELECT p.tb012_cod_produto, p.tb012_descricao, sum((v.tb010_012_quantidade * v.tb010_012_valor_unitario))/sum(v.tb010_012_quantidade) AS preco_medio_venda FROM staging.vendas v, staging.produtos p GROUP BY p.tb012_cod_produto, v.tb012_cod_produto, p.tb012_descricao HAVING v.tb012_cod_produto = p.tb012_cod_produto) AS subquery WHERE p.tb012_cod_produto = subquery.tb012_cod_produto

```
CREATE SCHEMA dw;
```

Name	Value
Updated Rows	0
Query	CREATE SCHEMA dw

```
CREATE TABLE dw.Cliente (
  cpf BIGINT PRIMARY KEY,
  nome VARCHAR
);
```

Name	Value
Updated Rows	0
Query	CREATE TABLE dw.Cliente (cpf BIGINT PRIMARY KEY, nome VARCHAR)

CREATE TABLE dw.Ano (

ano INTEGER PRIMARY KEY

);

Statistics 1 X

Name	Value
Updated Rows	0
Query	CREATE TABLE dw.Ano (
	ano INTEGER PRIMARY KEY
)

CREATE TABLE dw.Mes (

mes INTEGER PRIMARY KEY

);

Statistics 1 X

Name	Value
Updated Rows	0
Query	CREATE TABLE dw.Mes (
	mes INTEGER PRIMARY KEY
)

CREATE TABLE dw.Produto (

id INTEGER PRIMARY KEY,

descricao VARCHAR,

preco_medio_compra FLOAT,

preco_medio_venda FLOAT

);

Statistics 1 X

Name	Value
Updated Rows	0
Query	CREATE TABLE dw.Produto (
	id INTEGER PRIMARY KEY,
	descricao VARCHAR,
	preco_medio_compra FLOAT,
	preco_medio_venda FLOAT
)

CREATE TABLE dw.Categoria (
 id INTEGER PRIMARY KEY,
 descricao VARCHAR
);

Statistics 1 X

Name	Value
Updated Rows	0
Query	CREATE TABLE dw.Categoria (id INTEGER PRIMARY KEY, descricao VARCHAR)

CREATE TABLE dw.Dia (
 dia INTEGER PRIMARY KEY
);

Statistics 1 X

Name	Value
Updated Rows	0
Query	CREATE TABLE dw.Dia (dia INTEGER PRIMARY KEY)

CREATE TABLE dw.Tipo (
 id INTEGER PRIMARY KEY,
 descricao VARCHAR
);

Statistics 1 X

Name	Value
Updated Rows	0
Query	CREATE TABLE dw.Tipo (id INTEGER PRIMARY KEY, descricao VARCHAR)

SQL Query Editor:

```
CREATE TABLE dw.Venda (
    quantidade INTEGER,
    valor FLOAT,
    dia INTEGER,
    mes INTEGER,
    ano INTEGER,
    produto INTEGER,
    categoria INTEGER,
    tipo INTEGER,
    cliente BIGINT
);
```

Statistics 1

Name	Value
Updated Rows	0
Query	CREATE TABLE dw.Venda (quantidade INTEGER, valor FLOAT, dia INTEGER, mes INTEGER, ano INTEGER, produto INTEGER, categoria INTEGER, tipo INTEGER, cliente BIGINT)

SQL Query Editor:

```
CREATE TABLE dw.Compra (
    quantidade INTEGER,
    valor FLOAT,
    mes INTEGER,
    ano INTEGER,
    produto INTEGER
);
```

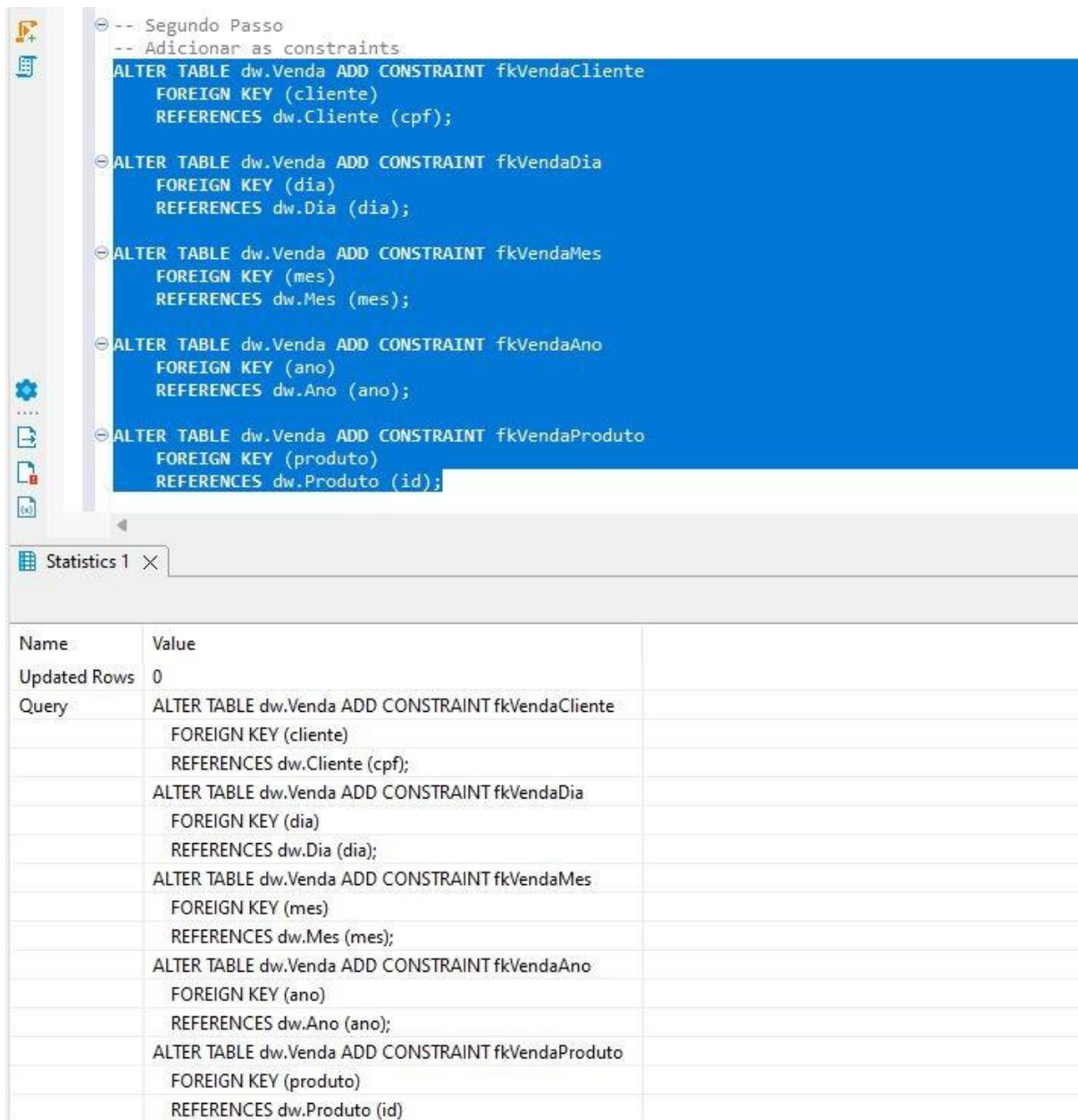
Statistics 1

Name	Value
Updated Rows	0
Query	CREATE TABLE dw.Compra (quantidade INTEGER, valor FLOAT, mes INTEGER, ano INTEGER, produto INTEGER)

```
CREATE TABLE dw.Lucro (
    produto INTEGER,
    quantidade INTEGER,
    lucro FLOAT
);
```

Statistics 1

Name	Value
Updated Rows	0
Query	CREATE TABLE dw.Lucro (produto INTEGER, quantidade INTEGER, lucro FLOAT)



-- Inserção de dados na tabela dimensão Ano

```
INSERT INTO dw.ano
  SELECT DISTINCT EXTRACT(YEAR FROM tb010_012_data) FROM staging.vendas
  ON CONFLICT DO NOTHING;
INSERT INTO dw.ano
  SELECT DISTINCT EXTRACT(YEAR FROM tb012_017_data) FROM staging.compras
  ON CONFLICT DO NOTHING;
```

Statistics 1 X

Name	Value
Updated Rows	1
Query	INSERT INTO dw.ano SELECT DISTINCT EXTRACT(YEAR FROM tb010_012_data) FROM staging.vendas ON CONFLICT DO NOTHING; INSERT INTO dw.ano SELECT DISTINCT EXTRACT(YEAR FROM tb012_017_data) FROM staging.compras ON CONFLICT DO NOTHING

-- Inserção de dados na tabela dimensão Mes

```
INSERT INTO dw.mes
  SELECT DISTINCT EXTRACT(MONTH FROM tb010_012_data) FROM staging.vendas
  ON CONFLICT DO NOTHING;
INSERT INTO dw.mes
  SELECT DISTINCT EXTRACT(MONTH FROM tb012_017_data) FROM staging.compras
  ON CONFLICT DO NOTHING;
```

Statistics 1 X

Name	Value
Updated Rows	12
Query	INSERT INTO dw.mes SELECT DISTINCT EXTRACT(MONTH FROM tb010_012_data) FROM staging.vendas ON CONFLICT DO NOTHING; INSERT INTO dw.mes SELECT DISTINCT EXTRACT(MONTH FROM tb012_017_data) FROM staging.compras ON CONFLICT DO NOTHING

-- Inserção de dados na tabela dimensão Dia

```
INSERT INTO dw.dia
  SELECT DISTINCT EXTRACT(DAY FROM tb010_012_data) FROM staging.vendas
  ON CONFLICT DO NOTHING;
```

Statistics 1 X

Name	Value
Updated Rows	9
Query	INSERT INTO dw.dia SELECT DISTINCT EXTRACT(DAY FROM tb010_012_data) FROM staging.vendas ON CONFLICT DO NOTHING

<pre>-- Inserção de dados na tabela dimensão Produto INSERT INTO dw.produto SELECT DISTINCT p.tb012_cod_produto, p.tb012_descricao, p.preco_medio_compra, p.preco_medio_venda FROM staging.produtos p;</pre>	
Statistics 1 X	
Name	Value
Updated Rows	68
Query	INSERT INTO dw.produto SELECT DISTINCT p.tb012_cod_produto, p.tb012_descricao, p.preco_medio_compra, p.preco_medio_venda FROM staging.produtos p

<pre>-- Inserção de dados na tabela dimensão Cliente INSERT INTO dw.cliente SELECT DISTINCT c.tb010_cpf, c.tb010_nome FROM staging.clientes c;</pre>	
Statistics 1 X	
Name	Value
Updated Rows	51
Query	INSERT INTO dw.cliente SELECT DISTINCT c.tb010_cpf, c.tb010_nome FROM staging.clientes c

<pre>-- Inserção de dados na tabela dimensão Categoria INSERT INTO dw.categoria SELECT DISTINCT c.tb013_cod_categoria, c.tb013_descricao FROM staging.categorias c;</pre>	
Statistics 1 X	
Name	Value
Updated Rows	8
Query	INSERT INTO dw.categoria SELECT DISTINCT c.tb013_cod_categoria, c.tb013_descricao FROM staging.categorias c

<pre>-- Inserção de dados na tabela dimensão Tipo INSERT INTO dw.tipo (id, descricao) VALUES (1, 'alimento'), (2, 'eletro'), (3, 'vestuario');</pre>	
Statistics 1 X	
Name	Value
Updated Rows	3
Query	INSERT INTO dw.tipo (id, descricao) VALUES (1, 'alimento'), (2, 'eletro'), (3, 'vestuario')

<div> <div> -- Inserção de dados na tabela fato Compra </div> <div> <pre> INSERT INTO dw.compra (produto, quantidade, valor, mes, ano) (SELECT c.tb012_cod_produto AS produto, c.tb012_017_quantidade AS quantidade, c.tb012_017_quantidade*c.tb012_017_valor_unitario AS valor, EXTRACT(MONTH FROM c.tb012_017_data) AS mes, EXTRACT(YEAR FROM c.tb012_017_data) AS ano FROM staging.compras c); </pre> </div> </div>		
Statistics 1		
Name	Value	
Updated Rows	38	
Query	INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)	
	(
	SELECT	
	c.tb012_cod_produto AS produto,	
	c.tb012_017_quantidade AS quantidade,	
	c.tb012_017_quantidade*c.tb012_017_valor_unitario AS valor,	
	EXTRACT(MONTH FROM c.tb012_017_data) AS mes,	
	EXTRACT(YEAR FROM c.tb012_017_data) AS ano	
	FROM	
	staging.compras c	
)	

```
-- Independente do mês
INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
  SELECT
    c.tb012_cod_produto AS produto,
    SUM(c.tb012_017_quantidade) AS quantidade,
    SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
    NULL AS mes,
    EXTRACT(YEAR FROM c.tb012_017_data) AS ano
  FROM
    staging.compras c
  GROUP BY c.tb012_cod_produto, EXTRACT(YEAR FROM c.tb012_017_data)
);
```

Statistics 1
×

Name	Value
Updated Rows	23
Query	<pre>-- Independente do mês INSERT INTO dw.compra (produto, quantidade, valor, mes, ano) (SELECT c.tb012_cod_produto AS produto, SUM(c.tb012_017_quantidade) AS quantidade, SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor, NULL AS mes, EXTRACT(YEAR FROM c.tb012_017_data) AS ano FROM staging.compras c GROUP BY c.tb012_cod_produto, EXTRACT(YEAR FROM c.tb012_017_data));</pre>

-- Independente do ano

```

INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
  SELECT
    c.tb012_cod_produto AS produto,
    SUM(c.tb012_017_quantidade) AS quantidade,
    SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
    EXTRACT(MONTH FROM c.tb012_017_data) AS mes,
    NULL AS ano
  FROM
    staging.compras c
  GROUP BY c.tb012_cod_produto, EXTRACT(MONTH FROM c.tb012_017_data)
);

```

Statistics 1 X

Name	Value
Updated Rows	38
Query	INSERT INTO dw.compra (produto, quantidade, valor, mes, ano) (SELECT c.tb012_cod_produto AS produto, SUM(c.tb012_017_quantidade) AS quantidade, SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor, EXTRACT(MONTH FROM c.tb012_017_data) AS mes, NULL AS ano FROM staging.compras c GROUP BY c.tb012_cod_produto, EXTRACT(MONTH FROM c.tb012_017_data))

<pre>-- Independente do ano e mês INSERT INTO dw.compra (produto, quantidade, valor, mes, ano) (SELECT c.tb012_cod_produto AS produto, SUM(c.tb012_017_quantidade) AS quantidade, SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor, NULL AS mes, NULL AS ano FROM staging.compras c GROUP BY c.tb012_cod_produto);</pre>		
Statistics 1 X		
Name	Value	
Updated Rows	23	
Query	INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)	
	(
	SELECT	
	c.tb012_cod_produto AS produto,	
	SUM(c.tb012_017_quantidade) AS quantidade,	
	SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,	
	NULL AS mes,	
	NULL AS ano	
	FROM	
	staging.compras c	
	GROUP BY c.tb012_cod_produto	
)	

-- Independente de produto
 INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
 (
 SELECT
 NULL AS produto,
 SUM(c.tb012_017_quantidade) AS quantidade,
 SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
 EXTRACT(MONTH FROM c.tb012_017_data) AS mes,
 EXTRACT(YEAR FROM c.tb012_017_data) AS ano
 FROM
 staging.compras c
 GROUP BY EXTRACT(MONTH FROM c.tb012_017_data), EXTRACT(YEAR FROM c.tb012_017_data)
);

Statistics 1 X

Name	Value
Updated Rows	12
Query	INSERT INTO dw.compra (produto, quantidade, valor, mes, ano) (SELECT NULL AS produto, SUM(c.tb012_017_quantidade) AS quantidade, SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor, EXTRACT(MONTH FROM c.tb012_017_data) AS mes, EXTRACT(YEAR FROM c.tb012_017_data) AS ano FROM staging.compras c GROUP BY EXTRACT(MONTH FROM c.tb012_017_data), EXTRACT(YEAR FROM c.tb012_017_data))

-- Independente do produto e mês

```

INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
  SELECT
    NULL AS produto,
    SUM(c.tb012_017_quantidade) AS quantidade,
    SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
    NULL AS mes,
    EXTRACT(YEAR FROM c.tb012_017_data) AS ano
  FROM
    staging.compras c
  GROUP BY EXTRACT(YEAR FROM c.tb012_017_data)
);

```

Statistics 1 X

Name	Value
Updated Rows	1
Query	INSERT INTO dw.compra (produto, quantidade, valor, mes, ano) (SELECT NULL AS produto, SUM(c.tb012_017_quantidade) AS quantidade, SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor, NULL AS mes, EXTRACT(YEAR FROM c.tb012_017_data) AS ano FROM staging.compras c GROUP BY EXTRACT(YEAR FROM c.tb012_017_data))

-- Independente do produto e ano
INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
SELECT
NULL AS produto,
SUM(c.tb012_017_quantidade) AS quantidade,
SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
EXTRACT(MONTH FROM c.tb012_017_data) AS mes,
NULL AS ano
FROM
staging.compras c
GROUP BY EXTRACT(MONTH FROM c.tb012_017_data)
);

Statistics 1

Name	Value
Updated Rows	12
Query	INSERT INTO dw.compra (produto, quantidade, valor, mes, ano) (SELECT NULL AS produto, SUM(c.tb012_017_quantidade) AS quantidade, SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor, EXTRACT(MONTH FROM c.tb012_017_data) AS mes, NULL AS ano FROM staging.compras c GROUP BY EXTRACT(MONTH FROM c.tb012_017_data))


```
-- Independente do produto, mês e ano
INSERT INTO dw.compra (produto, quantidade, valor, mes, ano)
(
  SELECT
    NULL AS produto,
    SUM(c.tb012_017_quantidade) AS quantidade,
    SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor,
    NULL AS mes,
    NULL AS ano
  FROM
    staging.compras c
);
```

Statistics 1

Name	Value
Updated Rows	1
Query	INSERT INTO dw.compra (produto, quantidade, valor, mes, ano) (SELECT NULL AS produto, SUM(c.tb012_017_quantidade) AS quantidade, SUM(c.tb012_017_quantidade*c.tb012_017_valor_unitario) AS valor, NULL AS mes, NULL AS ano FROM staging.compras c)

-- Inserção de dados na tabela fato Venda

```

INSERT INTO
dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo, cliente)
(
SELECT
v.tb010_012_quantidade AS quantidade,
v.tb010_012_quantidade*v.tb010_012_valor_unitario AS valor,
EXTRACT(DAY FROM v.tb010_012_data) AS dia,
EXTRACT(MONTH FROM v.tb010_012_data) AS mes,
EXTRACT(YEAR FROM v.tb010_012_data) AS ano,
v.tb012_cod_produto AS produto,
p.tb013_cod_categoria AS categoria,
p.tipo_produto AS tipo,
v.tb010_cpf AS cliente
FROM
staging.vendas v,
staging.produtos p
WHERE
v.tb012_cod_produto = p.tb012_cod_produto
);

```

Statistics 1

Name	Value
Updated Rows	32
Query	<pre> INSERT INTO dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo, cliente) (SELECT v.tb010_012_quantidade AS quantidade, v.tb010_012_quantidade*v.tb010_012_valor_unitario AS valor, EXTRACT(DAY FROM v.tb010_012_data) AS dia, EXTRACT(MONTH FROM v.tb010_012_data) AS mes, EXTRACT(YEAR FROM v.tb010_012_data) AS ano, v.tb012_cod_produto AS produto, p.tb013_cod_categoria AS categoria, p.tipo_produto AS tipo, v.tb010_cpf AS cliente FROM staging.vendas v, staging.produtos p WHERE v.tb012_cod_produto = p.tb012_cod_produto); </pre>

<pre>-- considerando categoria, tipo e quantidade INSERT INTO dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo, cliente) (SELECT SUM(v.tb010_012_quantidade) AS quantidade, NULL AS valor, NULL AS dia, NULL AS mes, NULL AS ano, NULL AS produto, p.tb013_cod_categoria AS categoria, p.tipo_produto AS tipo, NULL AS cliente FROM staging.vendas v, staging.produtos p WHERE v.tb012_cod_produto = p.tb012_cod_produto GROUP BY p.tb013_cod_categoria, p.tipo_produto);</pre>		
Statistics 1		
Name	Value	
Updated Rows	6	
Query	<pre>INSERT INTO dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo, cliente) (SELECT SUM(v.tb010_012_quantidade) AS quantidade, NULL AS valor, NULL AS dia, NULL AS mes, NULL AS ano, NULL AS produto, p.tb013_cod_categoria AS categoria, p.tipo_produto AS tipo, NULL AS cliente FROM staging.vendas v, staging.produtos p WHERE v.tb012_cod_produto = p.tb012_cod_produto GROUP BY p.tb013_cod_categoria, p.tipo_produto</pre>	

<pre>-- consideranco quantidade, valor, dia, mes, ano e cliente INSERT INTO dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo, cliente) (SELECT SUM(v.tb010_012_quantidade) AS quantidade, SUM(v.tb010_012_quantidade*v.tb010_012_valor_unitario) AS valor, EXTRACT(DAY FROM v.tb010_012_data) AS dia, EXTRACT(MONTH FROM v.tb010_012_data) AS mes, EXTRACT(YEAR FROM v.tb010_012_data) AS ano, NULL AS produto, NULL AS categoria, NULL AS tipo, v.tb010_cpf AS cliente FROM staging.vendas v GROUP BY EXTRACT(DAY FROM v.tb010_012_data), EXTRACT(MONTH FROM v.tb010_012_data), EXTRACT(YEAR FROM v.tb010_012_data), v.tb010_cpf);</pre>		
Statistics 1		
Name	Value	
Updated Rows	32	
Query	<pre>INSERT INTO dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo, cliente) (SELECT SUM(v.tb010_012_quantidade) AS quantidade, SUM(v.tb010_012_quantidade*v.tb010_012_valor_unitario) AS valor, EXTRACT(DAY FROM v.tb010_012_data) AS dia, EXTRACT(MONTH FROM v.tb010_012_data) AS mes, EXTRACT(YEAR FROM v.tb010_012_data) AS ano, NULL AS produto, NULL AS categoria, NULL AS tipo, v.tb010_cpf AS cliente FROM staging.vendas v GROUP BY EXTRACT(DAY FROM v.tb010_012_data), EXTRACT(MONTH FROM v.tb010_012_data), EXTRACT(YEAR FROM v.tb010_012_data), v.tb010_cpf);</pre>	

```
-- consideranco quantidade, valor, mes, ano e cliente
INSERT INTO
  dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo, cliente)
(
  SELECT
    SUM(v.tb010_012_quantidade) AS quantidade,
    SUM(v.tb010_012_quantidade*v.tb010_012_valor_unitario) AS valor,
    NULL AS dia,
    EXTRACT(MONTH FROM v.tb010_012_data) AS mes,
    EXTRACT(YEAR FROM v.tb010_012_data) AS ano,
    NULL AS produto,
    NULL AS categoria,
    NULL AS tipo,
    v.tb010_cpf AS cliente
  FROM
    staging.vendas v
  GROUP BY
    EXTRACT(MONTH FROM v.tb010_012_data),
    EXTRACT(YEAR FROM v.tb010_012_data),
    v.tb010_cpf
);
```

Statistics 1 X

Name	Value	
Updated Rows	24	
Query	INSERT INTO dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo, cliente) (SELECT SUM(v.tb010_012_quantidade) AS quantidade, SUM(v.tb010_012_quantidade*v.tb010_012_valor_unitario) AS valor, NULL AS dia, EXTRACT(MONTH FROM v.tb010_012_data) AS mes, EXTRACT(YEAR FROM v.tb010_012_data) AS ano, NULL AS produto, NULL AS categoria, NULL AS tipo, v.tb010_cpf AS cliente FROM staging.vendas v GROUP BY EXTRACT(MONTH FROM v.tb010_012_data), EXTRACT(YEAR FROM v.tb010_012_data), v.tb010_cpf))	

```
-- considerando quantidade, valor e cliente
INSERT INTO
  dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo, cliente)
(
  SELECT
    SUM(v.tb010_012_quantidade) AS quantidade,
    SUM(v.tb010_012_quantidade*v.tb010_012_valor_unitario) AS valor,
    NULL AS dia,
    NULL AS mes,
    NULL AS ano,
    NULL AS produto,
    NULL AS categoria,
    NULL AS tipo,
    v.tb010_cpf AS cliente
  FROM
    staging.vendas v
  GROUP BY
    v.tb010_cpf
);
```

Statistics 1		
Name	Value	
Updated Rows	24	
Query	INSERT INTO dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo, cliente) (SELECT SUM(v.tb010_012_quantidade) AS quantidade, SUM(v.tb010_012_quantidade*v.tb010_012_valor_unitario) AS valor, NULL AS dia, NULL AS mes, NULL AS ano, NULL AS produto, NULL AS categoria, NULL AS tipo, v.tb010_cpf AS cliente FROM staging.vendas v GROUP BY v.tb010_cpf)	


```

-- considerando quantidade, valor, ano e cliente
INSERT INTO
  dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo, cliente)
(
  SELECT
    SUM(v.tb010_012_quantidade) AS quantidade,
    SUM(v.tb010_012_quantidade*v.tb010_012_valor_unitario) AS valor,
    NULL AS dia,
    NULL AS mes,
    EXTRACT(YEAR FROM v.tb010_012_data) AS ano,
    NULL AS produto,
    NULL AS categoria,
    NULL AS tipo,
    v.tb010_cpf AS cliente
  FROM
    staging.vendas v
  GROUP BY
    EXTRACT(YEAR FROM v.tb010_012_data),
    v.tb010_cpf
);

```

Statistics 1		
Name	Value	
Updated Rows	24	
Query	INSERT INTO dw.venda (quantidade, valor, dia, mes, ano, produto, categoria, tipo, cliente) (SELECT SUM(v.tb010_012_quantidade) AS quantidade, SUM(v.tb010_012_quantidade*v.tb010_012_valor_unitario) AS valor, NULL AS dia, NULL AS mes, EXTRACT(YEAR FROM v.tb010_012_data) AS ano, NULL AS produto, NULL AS categoria, NULL AS tipo, v.tb010_cpf AS cliente FROM staging.vendas v GROUP BY EXTRACT(YEAR FROM v.tb010_012_data), v.tb010_cpf));	

<pre>-- Inserção de dados na tabela fato Lucro INSERT INTO dw.lucro (produto, quantidade, lucro) (SELECT v.tb012_cod_produto AS produto, SUM(v.tb010_012_quantidade) AS quantidade, SUM(v.tb010_012_quantidade * (p.preco_medio_venda - p.preco_medio_compra)) AS lucro FROM staging.vendas v, staging.produtos p WHERE v.tb012_cod_produto = p.tb012_cod_produto AND p.preco_medio_compra IS NOT NULL GROUP BY v.tb012_cod_produto);</pre>		
Statistics 1 X		
Name	Value	
Updated Rows	13	
Query	INSERT INTO dw.lucro (produto, quantidade, lucro)	
	(
	SELECT	
	v.tb012_cod_produto AS produto,	
	SUM(v.tb010_012_quantidade) AS quantidade,	
	SUM(v.tb010_012_quantidade * (p.preco_medio_venda - p.preco_medio_compra)) AS lucro	
	FROM	
	staging.vendas v,	
	staging.produtos p	
	WHERE	
	v.tb012_cod_produto = p.tb012_cod_produto AND	
	p.preco_medio_compra IS NOT NULL	
	GROUP BY	
	v.tb012_cod_produto	
)	

<pre>-- lucro total INSERT INTO dw.lucro (produto, quantidade, lucro) (SELECT NULL AS produto, NULL AS quantidade, SUM(v.tb010_012_quantidade * (p.preco_medio_venda - p.preco_medio_compra)) AS lucro FROM staging.vendas v, staging.produtos p WHERE v.tb012_cod_produto = p.tb012_cod_produto AND p.preco_medio_compra IS NOT NULL);</pre>		
Statistics 1 X		
Name	Value	
Updated Rows	1	
Query	INSERT INTO dw.lucro (produto, quantidade, lucro)	
	(
	SELECT	
	NULL AS produto,	
	NULL AS quantidade,	
	SUM(v.tb010_012_quantidade * (p.preco_medio_venda - p.preco_medio_compra)) AS lucro	
	FROM	
	staging.vendas v,	
	staging.produtos p	
	WHERE	
	v.tb012_cod_produto = p.tb012_cod_produto AND	
	p.preco_medio_compra IS NOT NULL	
)	