



UNIVERSIDADE DO MINHO
**Mestrado Integrado em Engenharia
Informática**
Análise de Dados

Northwind Data Warehouse
Trabalho Prático

Grupo 07

João Gomes, A74033
Joel Morais, A70841
Miguel Cunha, A78478
Nadine Oliveira, A75614

20 de Janeiro de 2019

Resumo

Projeto no âmbito da Unidade Curricular Analise de Dados do 4ºAno do Mestrado Integrado em Engenharia Informática.

O objetivo é analisar, planear e implementar um Data Warehouse com base no modelo e Scripts da base de dados *Northwind*.

Este projeto tem a finalidade de aprofundar os conhecimentos adquiridos nas aulas, e ainda de nos mostrar as dificuldades encontradas no processo de implementação de um Data Warehouse.



Conteúdo

1	Introdução	3
1.1	Contextualização	3
1.2	Fundamentação do Sistema	3
2	Modelo Dimensional do Sistema	4
2.1	Definição e Caracterização das Dimensões	4
2.2	Definição e Caracterização das Tabelas de Factos	6
2.3	Esquematização do Esquema Dimensional	7
3	Caracterização das Fontes de Informação	8
3.1	Identificação e Caracterização das Fontes do Sistema	8
3.2	Análise dos Dados das Fontes	8
3.3	Desenvolvimento do Esquema de Mapeamento de Dados	8
4	Implementação do Sistema de Data Warehousing	9
4.1	Implementação do esquema físico do sistema de dados	9
4.2	Área de retenção	10
5	Indicadores de BI	13
5.1	Quantidade de cada produto	13
5.2	Preço base de cada produto e Preço de Venda de cada produto	13
5.3	Compradores por produto	14
5.4	Preço de venda por produto e por categoria	15
5.5	Quantidade de produtos encomendados por cargo	15
5.6	Quantidade de produtos enviados por nome e fornecedor	16
5.7	Quantidade de produtos enviados por empresa de entrega	17
	Conclusão	18

Lista de Tabelas

Lista de Figuras

1	Dimensão Shipper.	4
2	Dimensão Customer.	4
3	Dimensão Supplier.	5
4	Dimensão Products.	5
5	Dimensão Employee.	5
6	Dimensão Time.	6
7	Tabela de Factos Orders_Fact.	6
8	Tabela de Factos Purchase_Order_Fact.	6
9	Esquema do Data Warehouse.	7
10	Tabela Customer	10
11	Tabela Purchase Order Fact	11
12	Quantidade de cada produto	13



LISTA DE FIGURAS

13	Preço base de cada produto e Preço de Venda de cada produto	14
14	Compradores por produto	14
15	Preço de venda por produto e por categoria	15
16	Quantidade de produtos encomendados por cargo	16
17	Quantidade de produtos enviados por nome e fornecedor	16
18	Quantidade de produtos enviados por empresa de entrega	17



1 Introdução

1.1 Contextualização

Neste projeto foi-nos pedido que realizássemos todos os passos para a implementação de Data Warehouse para a base de dados *Northwind* fornecida, este processo de implementação contém a fase de planeamento, o processo ETL (Extract, Transform, Load) e uma fase de análise recorrendo à ferramenta *Power BI*.

A base de dados da *Northwind* suporta a operacionalidade de uma empresa fictícia que se dedica ao comércio internacional de produtos alimentares.

1.2 Fundamentação do Sistema

O Data Warehouse implementado, bem como, os indicadores de *BI* criados, vão ajudar na análise deste modelo de negócio tendo noção de certas particularidades ao apresentarmos um histórico da mesma, tal como o nível de pedidos por regiões, por clientes ou mesmo os que fornecem maior quantidade de produtos.

Pretendemos apresentar uma implementação capaz de tratar e permitir realizar uma boa análise de todos os dados do sistema, ajudando assim o modelo de negócio da *Northwind*, embora não se tratando de uma empresa real.



2 Modelo Dimensional do Sistema

De forma a implementar o nosso Data Warehouse corretamente tivemos de o planejar, e criar o nosso modelo Dimensional.

2.1 Definição e Caracterização das Dimensões

O nosso modelo dimensional, apresenta seis tabelas de dimensão que têm por base as tabelas da base de dados *Northwind*, retirando delas atributos que não achamos necessários para a análise efetuada pelo nosso sistema e colocando todos os valores como Not Null, sendo elas:

- **Dimensão *Shipper*:**

Esta dimensão apresenta atributos retirados da tabela *Shippers* sendo representada da seguinte forma:

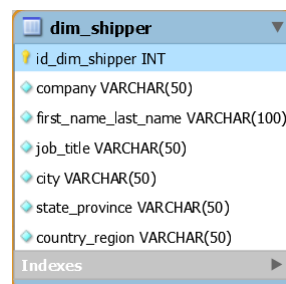


Figura 1: Dimensão Shipper.

- **Dimensão *Customer*:**

Os atributos desta dimensão estão presentes na tabela *Customer*, sendo representada da seguinte forma:

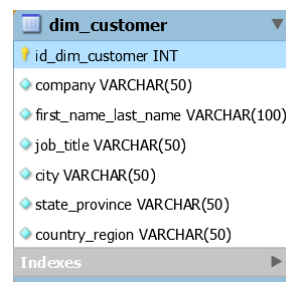


Figura 2: Dimensão Customer.



- **Dimensão *Supplier*:**

Nesta dimensão juntamos os atributos da tabela *Supplier*, como mostra a figura:

dim_supplier	
id_dim_supplier	INT
company	VARCHAR(50)
first_name_last_name	VARCHAR(100)
job_title	VARCHAR(50)
city	VARCHAR(50)
state_province	VARCHAR(50)
country_region	VARCHAR(50)

Figura 3: Dimensão Supplier.

- **Dimensão *Products*:**

Na tabela *Products* encontramos os atributos desta dimensão:

dim_products	
id_dim_products	INT
name	VARCHAR(50)
standard_cost	DECIMAL(19,4)
list_price	DECIMAL(19,4)
quantity_per_unit	VARCHAR(50)
discontinued	TINYINT(1)
category	VARCHAR(50)

Figura 4: Dimensão Products.

- **Dimensão *Employee*:**

Esta dimensão apresenta os atributos retirados da tabela *Employee*, tal como mostra a figura:

dim_employee	
id_dim_employer	INT
company	VARCHAR(50)
first_name_last_name	VARCHAR(100)
job_title	VARCHAR(50)
city	VARCHAR(50)
state_province	VARCHAR(50)
country_region	VARCHAR(50)

Figura 5: Dimensão Employee.

- **Dimensão *Time*:**

Esta dimensão não tem origem numa tabela, mas sim nos atributos de várias tabelas, sendo eles *order_date*, *paid_date*, *shipped_date*, *submitted_date*, *payment_date*, *approved_date*, criando assim a seguinte tabela:

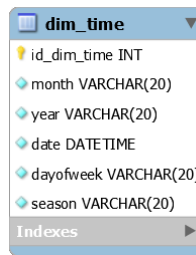


Figura 6: Dimensão Time.

2.2 Definição e Caracterização das Tabelas de Factos

Para a criação das tabelas de factos tivemos de definir que factos iríamos considerar, para futura avaliação, ou seja atributos que não sofrem alterações ao longo do tempo, para isto definimos duas tabelas de factos, sendo elas as seguintes:

- **Orders_Fact:**

Esta tabela além das chaves estrangeiras que possui, tem ainda os atributos *unit_price* e *quantity* relativos a tabela *Order_Details*, como se pode observar na figura:

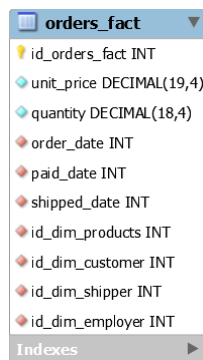


Figura 7: Tabela de Factos Orders_Fact.

- **Purchase_Order_Fact:**

Esta tabela além das chaves estrangeiras, têm ainda os atributos da tabela de cima mas relativos a tabela *Purchase_Order_Details*, como se pode observar:

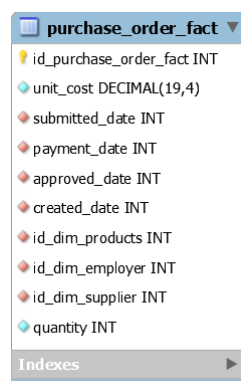


Figura 8: Tabela de Factos Purchase_Order_Fact.



2.3 Esquematização do Esquema Dimensional

Depois de definidas as tabelas de dimensão e de Factos, partimos para a definição do nosso esquema dimensional, usamos um esquema em Constelação de Factos, isto é, possuímos tabelas de dimensão partilhadas pelas duas tabelas de factos criadas.

As tabelas partilhadas são Dim_Products, Dim_Employee e Dim_Time.

Optamos por esta solução devido a dimensão da fonte de dados, de modo a possuímos consultas simples e um povoamento mais intuitivo, mantendo uma boa capacidade de análise, assim conseguimos ter todas as vantagens que um esquema em estrela apresenta em relação ao esquema em floco de neve, mas com uma organização mais eficiente. Este é o esquema resultante do nosso planeamento:

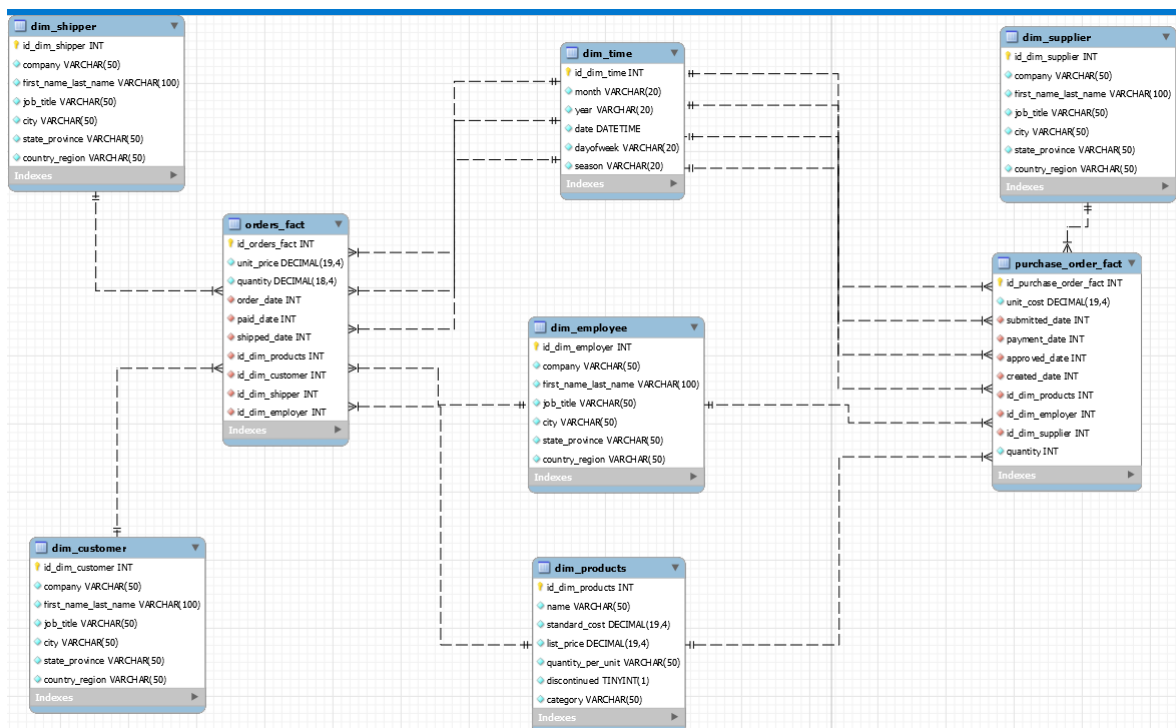


Figura 9: Esquema do Data Warehouse.



3 Caracterização das Fontes de Informação

Para este projeto vamos apenas usar uma fonte de dados, o *Northwind*.

3.1 Identificação e Caracterização das Fontes do Sistema

A *Northwind* é uma base de dados usada pela Microsoft para demonstrar algumas funcionalidades dos seus programas. Esta base de dados é constituída por dados referentes a vendas de companhias imaginárias, chamadas de *Northwind Traders*. A base de dados *Northwind* é constituída por várias tabelas (suppliers, employees, products, orders, customers, shippers, etc...). Estas tabelas serão povoadas pela *NorthWind* e iremos usar esses valores para o nosso Data Warehouse.

3.2 Análise dos Dados das Fontes

A base de dados *Northwind* foi criada com fins demonstrativos. Assim, a base de dados é bastante simples e os dados, em geral, são "bons". Contém alguns NULL's no entanto, mas ao tratar os dados iremos substituir esses valores por N/A, logo não irão apresentar qualquer tipo de problemas para o nosso Data Warehouse.

3.3 Desenvolvimento do Esquema de Mapeamento de Dados

O mapa lógico de dados é uma tabela que nos ajuda a entender e a expressar de onde e para onde vão os dados. Esta tabela guarda as tabelas, colunas, tipo de dados e tipo de tabelas onde os dados serão inseridos e de onde foram retirados. No final, cada linha apresenta uma coluna *transformations* que guardam notas e a maneira de como os dados foram carregados.



4 Implementação do Sistema de Data Warehousing

Nesta secção apresentamos todos os passos da implementação do esquema físico.

4.1 Implementação do esquema físico do sistema de dados

Depois de criarmos todas as tabelas do Data Warehouse, criamos procedimentos de modo a realizar o carregamento inicial dos dados com cursores em MySQL, segue em exemplo um desses procedimentos

```
DELIMITER $$
CREATE DEFINER='root'@'localhost' PROCEDURE '1_curCustomer'()
begin
    declare done int default false;
    declare aux_id_dim_customer int;
    declare aux_company varchar(50);
    declare aux_first_name_last_name varchar(100);
    declare aux_job_title varchar(50);
    declare aux_city varchar(50);
    declare aux_state_province varchar(50);
    declare aux_country_region varchar(50);
    declare cur cursor for select distinct id,company,concat(first_name,'
        ',last_name) as
        first_name_last_name,job_title,city,state_province,country_region from
        northwind.customers;
    declare continue handler for not found set done=true;
    open cur;
insert into
    trabalho.dim_customer(id_dim_customer,company,first_name_last_name,job_title,city,
state_province,country_region)
values(0,"N/A","N/A","N/A","N/A","N/A");
readloop: loop
fetch cur into
    aux_id_dim_customer,aux_company,aux_first_name_last_name,aux_job_title,aux_city,
aux_state_province,aux_country_region;
if aux_company is null then
    set aux_company = "N/A";
end if;
if aux_first_name_last_name is null then
    set aux_first_name_last_name = "N/A";
end if;
if aux_job_title is null then
    set aux_job_title = "N/A";
end if;
if aux_city is null then
    set aux_city = "N/A";
end if;
if aux_state_province is null then
    set aux_state_province = "N/A";
end if;
if aux_country_region is null then
    set aux_country_region = "N/A";
end if;
if done then
```



```
        leave readloop;
    end if;
    insert into
        trabalho.dim_customer(id_dim_customer,company,first_name_last_name,job_title,city,
        state_province,country_region)
    values(aux_id_dim_customer,aux_company,aux_first_name_last_name,aux_job_title,aux_city,
        aux_state_province,aux_country_region);
    end loop;
    close cur;
end$$
DELIMITER ;
```


Depois de termos este carregamento inicial, tivemos de criar uma Área de Retenção explicada em seguida.

4.2 Área de retenção

A Área de Retenção é uma área de armazenamento intermédio situada dentro do processo de ETL. Auxilia a transição dos dados das origens para o destino final no Data Warehouse.

No nosso caso, a nossa área de retenção apresenta uma tabela de auditoria para cada dimensão criada, e para cada tabela de factos. Como mostramos em seguida dois exemplos das nossas tabelas de auditoria:

- **Tabela de Auditoria *Customer*:**



aud_dim_customer	
id_aud_dim_customer	INT(11)
company	VARCHAR(50)
first_last_name	VARCHAR(100)
job_title	VARCHAR(50)
city	VARCHAR(50)
state_province	VARCHAR(50)
country_region	VARCHAR(50)
operation	VARCHAR(50)
Indexes	

Figura 10: Tabela Customer



- Tabela de Auditoria *Purchase_Order_Fact*:

Column Name	Data Type
idpre_order_facts	INT
unit_cost	DECIMAL(18,4)
quantity	DECIMAL(18,4)
submitted_date	INT
payment_date	INT
created_date	INT
approved_date	INT
id_dim_products	INT
id_dim_employer	INT
id_dim_supplier	INT
operation	VARCHAR(50)

Figura 11: Tabela Purchase Order Fact

De modo a reter os dados antes de atualizar o Data Warehouse criamos Triggers na fonte de dados, de modo a detetar qualquer inserção ou atualização da fonte de dados, de modo a atualizarmos o nosso Data Warehouse, criamos ainda Triggers na área de retenção, e assim possuímos total controlo sobre as operações utilizadas, bem como possuímos sempre o nosso Data Warehouse atualizado.

Exemplo Trigger na Fonte de Dados:

```
DROP TRIGGER IF EXISTS trigger_insert_customers;
DELIMITER $$
CREATE TRIGGER trigger_insert_customers
AFTER INSERT ON 'northwind'.customers
FOR EACH ROW
BEGIN
    DECLARE id INT;
    DECLARE op,cp,jt,ct,sp,cr VARCHAR(50);
    DECLARE fln VARCHAR(100);
    SET op = 'INSERT',
        id = new.id;
    IF(new.company is null) THEN SET cp = "N/A";
    ELSE SET cp = new.company;
    END IF;
    IF(new.first_name is null AND new.last_name is null) THEN SET fln = "N/A";
    ELSE SET fln = concat(new.first_name,' ', new.last_name);
    END IF;
    IF(new.job_title is null) THEN SET jt = "N/A";
    ELSE SET jt = new.job_title;
    END IF;
    IF(new.city is null) THEN SET ct = "N/A";
    ELSE SET ct = new.city;
    END IF;
    IF(new.state_province is null) THEN SET sp = "N/A";
    ELSE SET sp = new.state_province;
    END IF;
    IF(new.country_region is null) THEN SET cr = "N/A";
```



```
ELSE SET cr = new.country_region;
END IF;
INSERT INTO 'trabalho-ar'.aud_dim_customer values(id,cp,fln,jt,ct,sp,cr,op);
END$$
DELIMITER ;
```

Assim, sempre que novos dados forem inseridos na fonte, estes vão ser registados nas tabelas de auditoria, bem como a operação que foi efetuada (Insert ou Update). Isto, vai permitir manter os dados do data warehouse actualizados, pois, quando os dados são inseridos nas tabelas de auditoria, é verificada qual operação foi feita na fonte, e os dados no *data warehouse* são alterados em conformidade.

Para tal, foram criados triggers que detetavam quando nova informação era adicionada nas tabelas de auditoria e inseria a nova informação no data warehouse, nunca apagando a informação "desatualizada" deste, permitindo assim, manter um histórico da informação ao longo do tempo. Uma alternativa para manter os dados do data warehouse actualizados, seria com stored procedures, e o administrador da base de dados quando o processo de ETL fosse iniciado (periodicamente), eram verificados todos os dados nas tabelas de auditoria e atualizado o data warehouse. Porém, como este trabalho é em contexto académico, e uma dimensão bastante mais reduzida, optamos pela implementação com triggers, pois permite ter uma visualização mais rápida da evolução do data warehouse.

```
DELIMITER $$
CREATE TRIGGER trigger_aud_customers
AFTER INSERT ON 'trabalho-ar'.aud_dim_customer
FOR EACH ROW
BEGIN
    DECLARE last_id INT;
    IF(new.operation like 'INSERT') THEN
        INSERT INTO trabalho.dim_customer
        VALUES(new.idaud_dim_customer,new.company,new.first_last_name,new.job_title,
        new.city,new.state_province,new.country_region);
    END IF;
    IF(new.operation like 'UPDATE') THEN
        SET last_id = (SELECT id_dim_customer+1 FROM trabalho.dim_customer ORDER
        BY id_dim_customer DESC LIMIT 1);
        INSERT INTO trabalho.dim_customer
        VALUES(last_id,new.company,new.first_last_name,new.job_title,
        new.city,new.state_province,new.country_region);
    END IF;
END$$
DELIMITER ;
```



5 Indicadores de BI

Por último, e já com a implementação terminada, temos de passar à criação de diversos indicadores referentes ao nosso modelo dimensional, para isso, foram criadas diferentes dashboards com o intuito de fazer uma análise mais cuidada de cada indicador, podendo servir propósitos diferentes, como gerir material, ou simplesmente para fins estatísticos.

5.1 Quantidade de cada produto

Com este indicador podemos observar a quantidade relativa a cada produto da Northwind, optámos por utilizar um gráfico circular, visto que iria facilitar a análise de quantidades. Por exemplo, para o produto *Northwind Traders Beer* podemos ver que a sua quantidade é de 487, que representa 19.05% de todos os produtos.

Com esta análise podemos comparar a disponibilidade que temos para cada produto, tendo como finalidade uma melhor gestão de stocks.

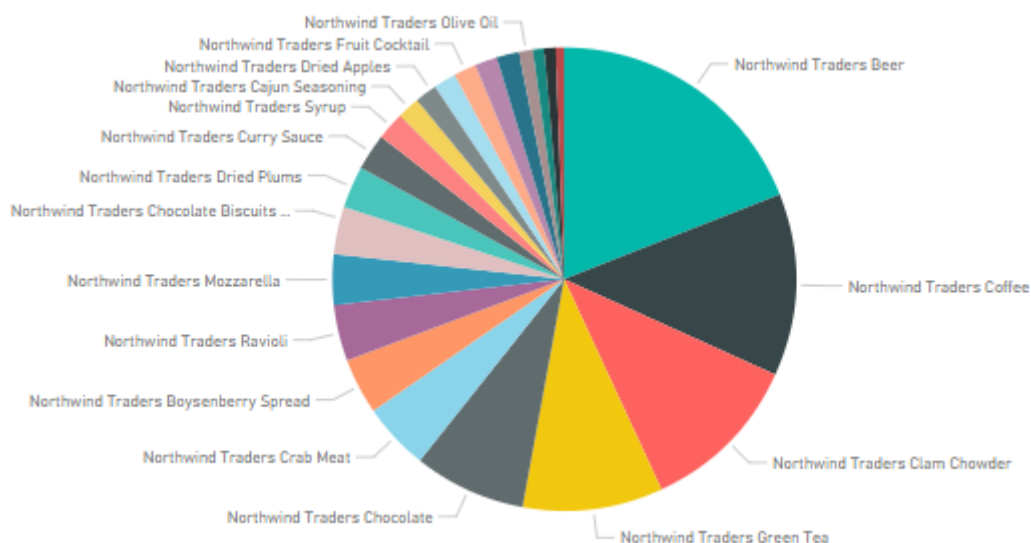


Figura 12: Quantidade de cada produto

5.2 Preço base de cada produto e Preço de Venda de cada produto

Estes dois indicadores têm como propósito mostrar o preço a que cada produto está a ser vendido, como mostrar o preço a que eles estão a ser comprados ao fornecedor. Podemos claramente observar que o produto *Northwind Traders Chai* está a ser adquirido a 13.5, no entanto está a ser vendido a 18.0. Optámos por utilizar um gráfico de linhas e por colocar estes dois indicadores na mesma dashboard, visto que a sua análise vai estar interligada. O objetivo principal é podermos observar o lucro que estamos a ter com cada produto, caso os preços do fornecedor sejam alterados, é possível observar também se o lucro aumentou ou diminuiu.



Figura 13: Preço base de cada produto e Preço de Venda de cada produto

5.3 Compradores por produto

Este indicador é representado por um Mapa de Manchas, que irá mostrar as regiões onde estão a ser comprados determinados produtos. Se tomarmos em atenção a cidade de New York, podemos observar que está a ser comprado o produto *Northwind Traders Chocolate Biscuits Mix* representada por uma mancha vermelha, e legendada pela cor vermelho. Através da observação dos compradores de certos produtos é possível fazer diversas análises, como os países que um certo produto está a atingir, se é necessário mudar portes de envio vendo a distâncias, entre outras.

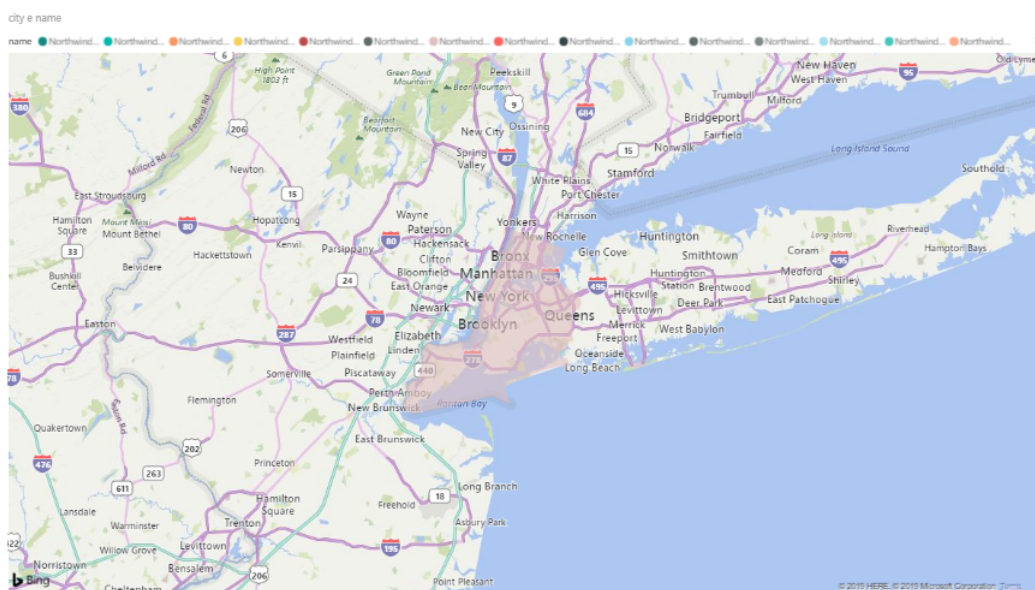


Figura 14: Compradores por produto



5.4 Preço de venda por produto e por categoria

Com este indicador podemos observar o preço de venda de cada produto, juntamente com a categoria a que ele pertence. Por exemplo, através das legendas por cores, podemos observar que dentro da categoria *Pasta* temos os produtos *Northwind Traders Ravioli* e *Northwind Traders Gnocchi* com os preços de 19.50 e 38.00 respetivamente.

Com este gráfico de colunas empilhadas é possível observar quais os produtos mais caros, mas também inspecionar a relação entre categorias, se os produtos dentro da mesma categoria apresentam preços semelhantes, ou mesmo se existir um interesse numa só categoria analisamos apenas essa.

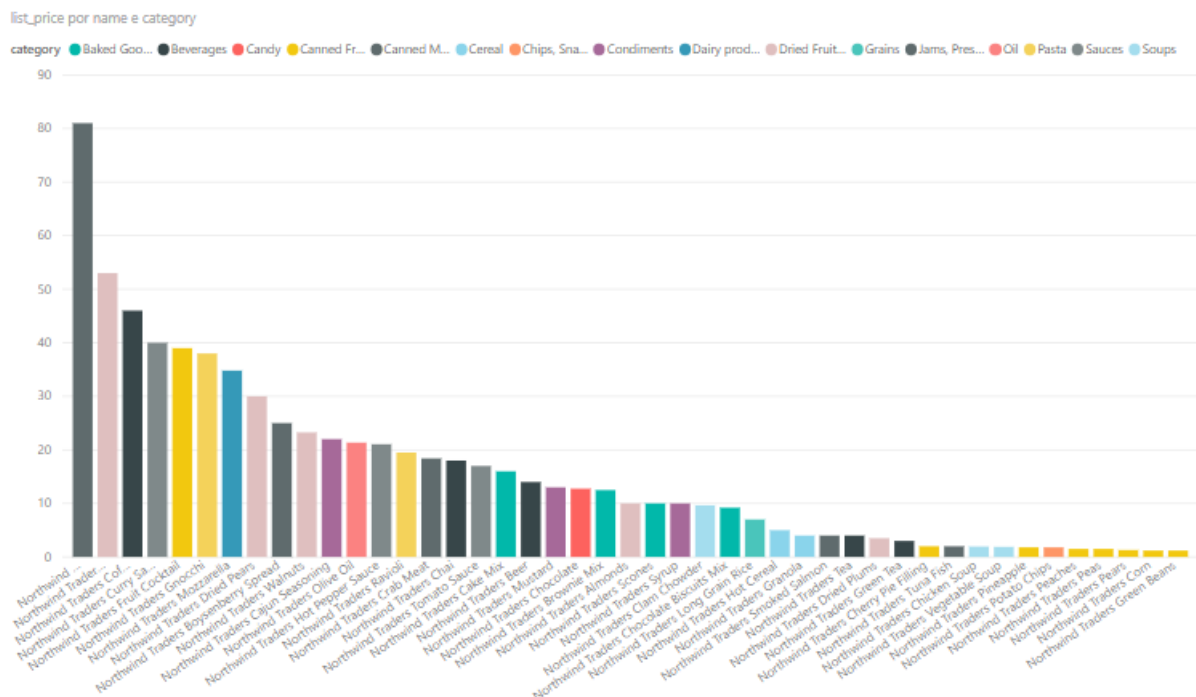


Figura 15: Preço de venda por produto e por categoria

5.5 Quantidade de produtos encomendados por cargo

Após a soma das datas de encomendas (que representa o número total de encomendas) conseguimos observar a quantidade que cada comprador (diferenciados pelo seu cargo) está a encomendar. Por exemplo, se um comprador dentro de uma empresa tiver o cargo de *Purchasing Manager*, este realizou 2954 encomendas, enquanto que um *Owner* só realizou 132 encomendas.

Este indicador serve principalmente para fins estatísticos.



order_date por job_title

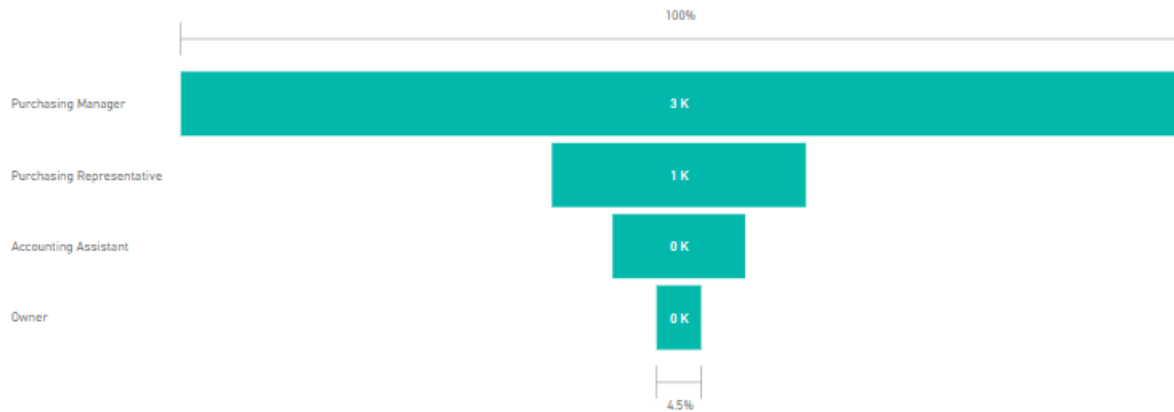


Figura 16: Quantidade de produtos encomendados por cargo

5.6 Quantidade de produtos enviados por nome e fornecedor

Através deste indicador podemos observar a quantidade de produtos que estão a ser enviado por cada fornecedor. Sabemos que o fornecedor B envia 474 do produto *Northwind Traders Chocolate*, no entanto só envia 83 do produto *Northwind Traders Scones*.

É possível com esta análise observar qual a empresa que envia determinado produto, ou ver ao contrário, uma certa empresa que quantidade de produtos envia.

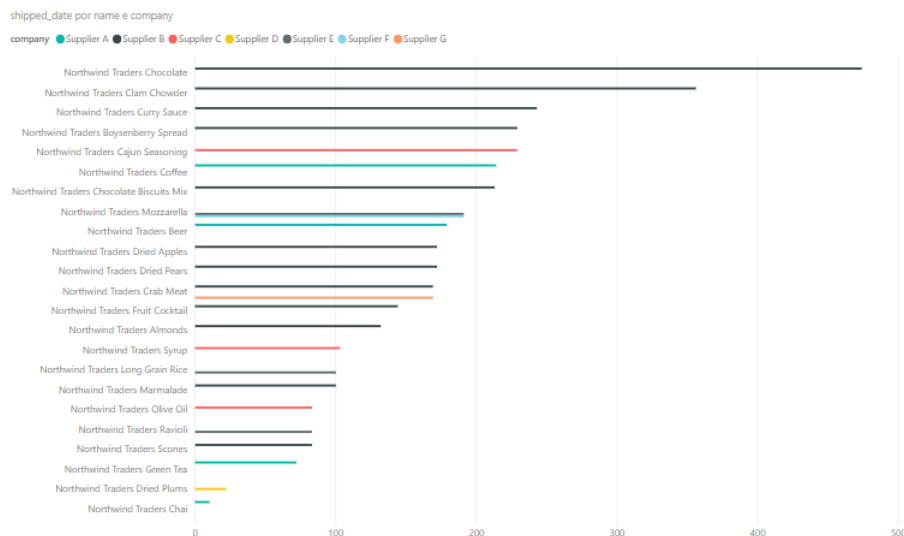


Figura 17: Quantidade de produtos enviados por nome e fornecedor



5.7 Quantidade de produtos enviados por empresa de entrega

Optámos por usar um gráfico em anel visto que a quantidade de empresas de entrega é reduzida, com isto podemos observar quantos produtos é que uma determinada empresa enviou, sabemos que a empresa A enviou 1023 produtos, a B 1350 e a C 1400 produtos.

Está análise tem principalmente fins estatísticos. Podendo observar qual das empresas enviou mais produtos na totalidade.

shipped_date por company

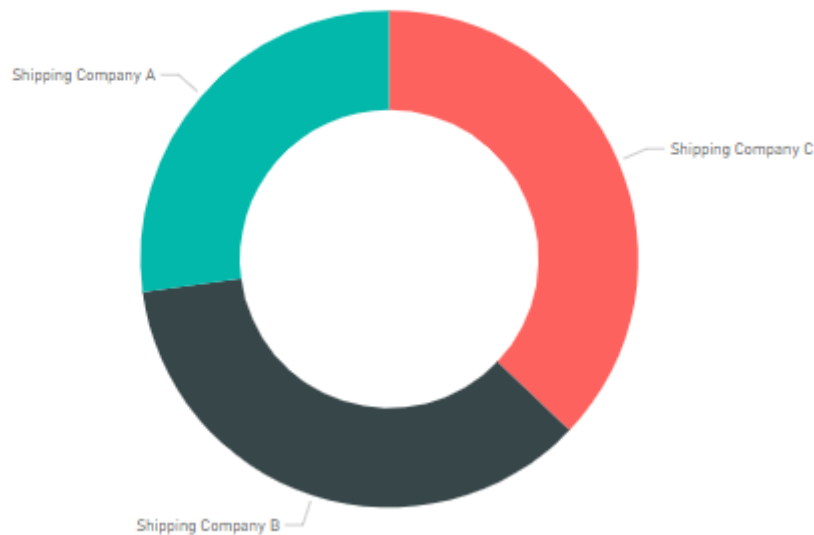


Figura 18: Quantidade de produtos enviados por empresa de entrega



Conclusão

Ao longo deste trabalho, e com os conhecimentos adquiridos nas diversas plataformas usadas (*MySQL* e o *PowerBi*), conseguimos concluir o mesmo com sucesso, mesmo que com algumas dificuldades pelo caminho.

Conseguimos entender que é muito importante prestar atenção aos dados que estamos a inserir no nosso sistema, pois nem todos os dados são relevantes e alguns deles podem ser considerados dados "maus". Com isto foi necessário recorrer a mecanismo de tratamento de dados.

Em suma, o resultado foi bastante positivo, pois conseguimos desenvolver um Data Warehouse, baseado na base de dados *NorthWind*, com sucesso e pronta para ser aplicada no mundo e ampliamos o nosso conhecimento no mundo de *Business Intelligence*, que futuramente, poderá ser bastante útil.