DATABASES PROJECT
(PROJET DE BASES DE DONNÉES)

Title of the report: ERP SALES OF “SUNOLTA”

Names of group members:
TIMUR KISHUBAIEV
ERULAN IBRAIMOV
VERICA DIMITROVA
ZIED ELLOUZI

Date: 25.05.2024

SELENA BASET
BASET ROMAIN

Table of Contents

1. Introduction

Context:

The plan of our project began with the idea of Timur Kishubaiev. He has already worked in a company called “Sunolta” for some years and he said that it would be useful to create a custom ERP system for registering the wholesale company's processes and providing advanced and flexible analytics. His company is currently using different software and accounting systems to store data but its functionality is limited and requires customisation. As far as ERP system is a very complex task that usually consists of some number of modules - each module for its more strict goal, and we were limited in time and our project size, it was a collective decision to try to implement a module ‘ERP sales’ that should be based on a MySQL database. ‘ERP sales’ was designed so it could be useful for every company that performs international wholesale trade.

Sunolta is a well-established Ukrainian company specializing in the processing and export of vegetable oils and their by-products. Founded in 1996, the company has grown to become a significant player in the agricultural sector, exporting its products to over 40 countries. Sunolta's main products include sunflower oil, meal, and other oilseed products, which are essential commodities in the global food and feed industries.

Problematic:

From our research and data gathering on Sunolta, we've observed that the company's expansion over the years has made managing its operations increasingly complex. The company faces several key challenges: operational data is fragmented across various systems, making it difficult to obtain a unified view of the business. Additionally, the lack of a centralized system complicates tracking inventory and predicting demand, which can lead to overstocking or stockouts. Manual processes and the absence of real-time data further slow down decision-making and operations, which is critical in the export business. Lastly, generating accurate reports for compliance and stakeholders is challenging without a robust system, risking non-compliance and reduced transparency. These challenges highlight the need for a comprehensive database management system to streamline operations and support continued growth.

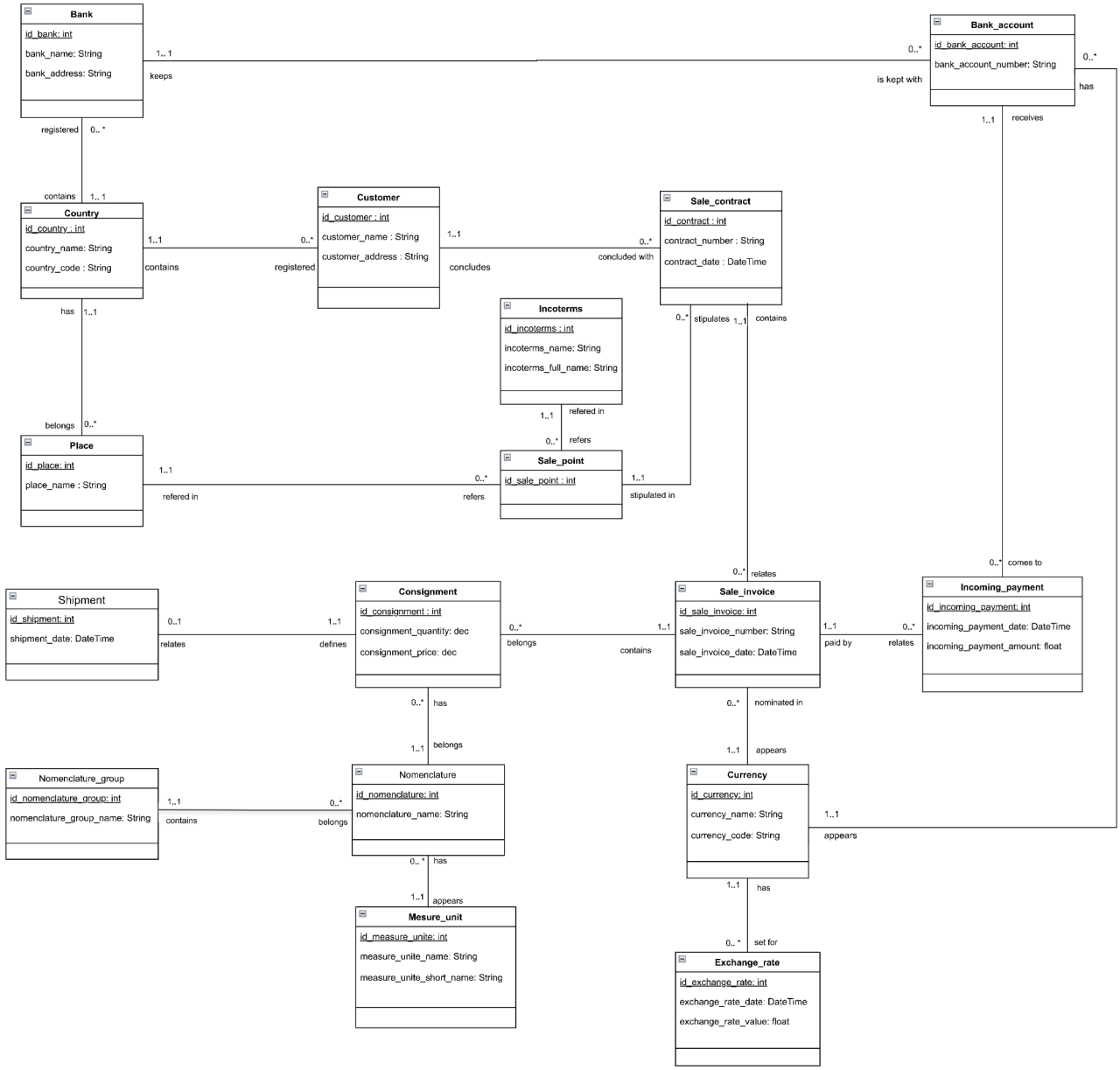
The data on the sales should be carried in different currencies, the prices should be analyzed with the reference to the sale terms, financial transactions should be carried through several bank accounts in different banks, countries and currencies, the moment of sale (date of the sale_invoice) could differ from the moment of shipment, and also the payment of the invoice may be performed partially and in different times. We have put these tasks to the ground of our further solution.

Solution:

The solution that we implement is a database in MySQL that allows us to store in a structured and normalized way the data on a company's sales. We decided to use fictional data in order to protect the private information of the company. The most part of data has been generated using randomness but within the logics of the underlying relations to a real world. The real company’s data could be inserted into this database structure and then all the analytics made on fictional data may be easily re-applied to the real data only by changing the content of the file ‘Hackers_donnees.sql’. Analytics has been performed by developing advanced SQL requests (views) and using them in Tableau Desktop to visualize the state of the company’s sales from different perspectives to make company’s management more informed while making their decisions.

2. Database Models

The following Class Diagram shows the Conceptual Model of our database:

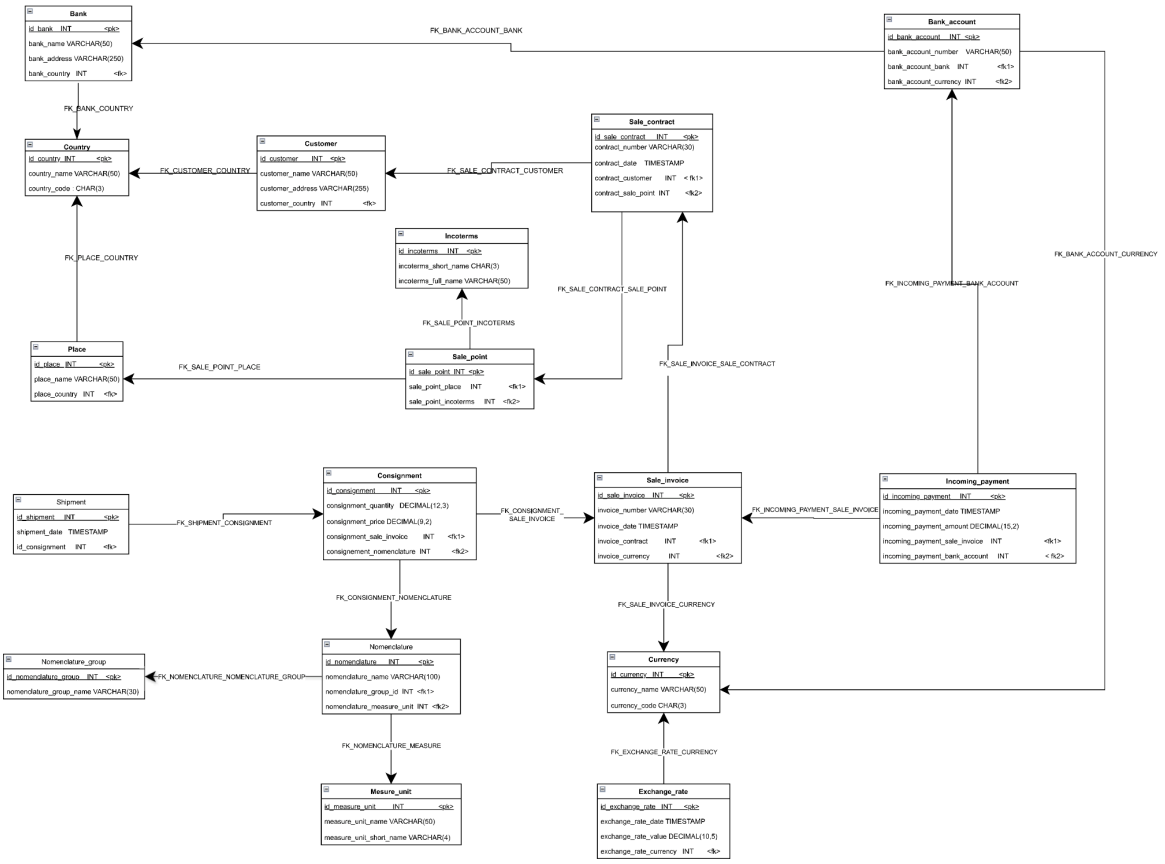


Detailed explanation of classes and their interaction (Fields we will describe more in relational model description):

1. Country - class reused in 3 other classes to identify the country of customer, bank and place of sale. Each country may be used by several customers, banks and places, but each of them has only one country to which it belongs. Very useful for further geographical visualizations.
2. Customer - class helping us to attach each sale to a certain customer and then perform analytics of each customer's performance and value for our company. Each customer may have several contracts, but each contract belongs only to one customer.
3. Place - class that means the point where we are selling the goods. It is used in class sale_point where together with Incoterms it fully identifies the place and terms of sale.
4. Incoterms - class that holds a dictionary of all standard terms of sale from Incoterms 2020.

5. Sale_point - class that identifies both place and terms of sale. Each sale_point is used in several contracts, but each contract has only one sale_point. Such choice of relationship is made due to business logic of future possible improvements of our ERP where it could be possible to generate the text of the contracts, as far as in reality each sale term has its standard form of contract and we usually don't mix it.
6. Currency - class that is used in 2 other classes Sale_invoice and Bank_account. Each sale invoice has only one currency and each bank account has only one currency, however each currency may appear 0 to many times in different sale invoices or bank accounts.
7. Exchange_rate - class that bears historical information on official Swiss national bank exchange rates (https://www.snb.ch/en/the-snb/mandates-goals/statistics/statistics-pub/current_interest_exchange_rates#t000)of different currencies compared to CHF. Each Exchange_rate record means that starting from a certain date for the certain currency the rate of this currency to CHF will become as in the field 'exchange_rate_value'. This logic will be used in our calculations while transferring different currencies amounts to CHF (or other currency) for comparison.
8. Bank - the class that reflects the banks where our company has its bank accounts. We may have several bank accounts in the same bank, but each bank account should be opened in one and only one bank.
9. Bank_account - the class reflecting our different bank accounts. It will help mark incoming payments so financial management will be aware of which account they went to and in which currency.
10. Sale_contract - class that reflects the parameters of a real contract like date and number and also the relation between customer and sale invoice. Each contract may have several sale invoices issued (or none), but each sale invoice must have one and only one contract related.
11. Sale_invoice - is one of the key classes that (together with Consignment - a line in the invoice) reflects the fact of the sale. Its date is used as the date of sale, and the date and currency are used for currency exchange calculations on sales amounts. Each invoice may have many lines (Consignments) in it, but each consignment belongs to one and only one sale_invoice.
12. Nomenclature_group - class that reflects the grouping of goods and services that we sell.
13. Measure_unit - class that holds the dictionary of units of measure of the goods and services that we sell, for example: mt - metric tons, pcs - pieces etc, l - liters etc. Each merchandise or service is measured in some unique measure_unit. If we have two different measures used in real life for the same merchandise or service, we'd prefer to make two separate instances of nomenclature each measured in only one measure_unit.
14. Nomenclature - class that represents identification for the goods or services that we sell. Each nomenclature may be included in several consignments, however our convention is to have one and only one nomenclature in each consignment. If we require different nomenclature in the same invoice, we'd better add another consignment with the unique nomenclature.
15. Consignment - class that gives us atomic representation of each consignment of goods with its quantity and price.
16. Incoming_payment - class that represents payments for the goods sold. We collect and store information on the date and amount paid for a certain sale invoice, bank account involved for further financial analysis of money flows and work with debtors.
17. Shipment - class that reflects the fact of shipment for correspondent consignment. For each consignment it may be shipped or not, in case it is shipped the unique object of Shipment class appears stipulating the date of such shipment.

The following diagram shows our relational model:



The relational model follows the same structure as the class diagram. All of our relationships are of one to many (including zero) type with only one exemption of “shipment” where the relation to ‘consignment’ is 0-1 to 1. We used foreign keys to reflect all these relationships in our database. In all classes primary keys are named “id_classname”. The foreign keys are named following this structure “fk_classname_referencedclass”. Field types were chosen according following logic that reflects the schedule of the real data size and further use in analytics: for primary and foreign keys - INT, for dates - TIMESTAMP, for price: DECIMAL(9,2), for payments amounts DECIMAL(15,2), for quantities of goods DECIMAL(12,3), for contract_number, invoice_number, nomenclature_group_name - VARCHAR(30), for names - generally VARCHAR(50), nomenclature_name - VARCHAR(100) as it could be a lot of details in description, for country code, currency code and Incoterms short name - CHAR(3) as they all are standard 3-letter significations.

3. Database Scripting

We distributed our SQL scripts between 3 files:

1) ***Hackers_structure_bdd.sql*** : This file contains database structure initialization, namely:
   a) Creation of database (preceded by dropping it if it existed before for reusability of this code within any environment that may already have some version of our base)
   b) Creation of tables and adding of foreign key constraints in accordance with our relational model diagram above
   c) Creation of 3 triggers:
      i) *before_country_delete* - to ensure that deletion of the country will not hurt dependent records in tables ‘place’, ‘customer’ and ‘bank’. We know that such a check is performed by corresponding

foreign key constraints, but we would like to show custom error messages so the user may know more precisely where to search for existing references to this country.

ii) *before_incoming_payment_insert, before_incoming_payment_update* - two equivalent triggers to ensure that the currency of payment (that is determined by the correspondent bank account) is equal to the currency of the invoice.

2) ***Hackers_donnees.sql*** : This file contains all that relates to generation and insertion of the data, namely:

a) Manual insertion of data into tables 'country', 'customer', 'place', 'incoterms', 'sale_point', 'currency', 'sale_contract', 'bank', 'bank_account', 'exchange_rate', 'sale_invoice', 'measure_unit', 'nomenclature_group', 'nomenclature'

b) Insertion to the 'consignment' table of the data generated by the following algorithm:

i) We created supplementary table 'indicative_price' that keeps some 'real life' parameters for random price generation for each of our existing products:

(1) min_price, max_price - overall minimum and maximum for each product that the generated price should not overcome

(2) start_price - starting minimal estimation of price for each step of generation

(3) price_steps - range to a) form random generated price, b) shift the start_price of next iteration

ii) We made procedure 'add_random_consignment' that:

(1) takes as parameters sale_invoice, nomenclature, min_quantity, max_quantity, min_price, max_price

(2) generates random quantity and price within given ranges

(3) inserts into 'consignment' table the record with correspondent incoming and generated values

iii) We made procedure 'change_prices' that updates column 'start_price' in table 'indicative_prices'

iv) with new randomly generated values within +/- price_steps range from previous start_price, but also respecting min_price, max_price limitations

v) We made a supplementary table 'for_debugging' to write there values of internal variables to debug the following procedure 'generate_consignments' as far as it didn't work as expected 'from scratch' and needed some adjustments to be made to the algorithm to finally perform well.

vi) We made a procedure 'generate_consignments' that:

(1) takes as arguments min_quantity and max_quantity - the limits for random quantity generation, and also max_number_of_consignments that should be generated for each sale_invoice

(2) makes cycle through each of sale_invoice existing in database and for each of them adds random number (from 1 to max_number_of_consignments) of consignments, using 'add_random_consignment' and correspondent parameters, and then executes 'change_prices' and updates parameters for next invoice consignments generation

vii) We executed 'generate_consignments' with 'real life' parameters to insert data in table 'consignment'. Below you may see how the first records looked after insertion.

id_consignment	consignment_sale_invoice	consignment_nomenclature	consignment_quantity	consignment_price
1	1	5	3266.456	204.80
2	1	5	2598.177	208.92
3	2	4	5757.057	196.89
4	2	4	2880.765	201.03
5	3	5	4097.124	222.42
6	4	3	2867.816	198.59
7	5	5	2728.645	230.54
8	5	5	4631.817	233.05
9	6	5	2769.157	227.57
10	6	5	4035.126	231.74
11	6	5	6744.976	231.81
12	7	4	2627.449	199.56
13	8	2	5796.290	851.88
14	8	2	6168.770	871.86

c) Insertion to the 'shipments' table of the data generated by the following algorithm:
   i) We created procedure 'generate_shipments' that:
      (1) takes as parameters the maximum number of 'days_before' and 'days_after' the date of sale invoice, so the date of shipment may be generated within this interval
      (2) for each consignment in the database randomly generates 'shipment_date' within mentioned interval and inserts correspondent record to the table 'shipment' (we suppose that our company is a reliable partner and all goods that we sold were shipped within some time)))
   ii) We executed the procedure 'generate_shipments' with 'real life' parameters 10 days before and 20 days after. Here is the result (first records):

id_shipment	shipment_date	id_consignment
1	2023-01-31 10:00:00	1
2	2023-01-24 10:00:00	2
3	2023-01-24 11:00:00	3
4	2023-01-13 11:00:00	4
5	2023-01-22 12:00:00	5
6	2023-01-06 13:00:00	6

d) Creation of the function 'get_exchange_rate' that:
   i) takes as arguments 'currency_id' and 'target_timestamp'
   ii) searches for the record in 'exchange_rate' with the same currency_id and latest date before 'target_timestamp' and returns correspondent 'exchange_rate' in format DECIMAL(10,5)

e) Creation of the view 'sale_invoice_full_aggregated_view' that will be used in generation of incoming payments (by providing the amount unpaid for each invoice) and also slightly changed - in Tableau for the visualization table of unpaid invoices (a very useful feature for the finance department). This view was designed to work in situations even when sale contracts and/or customers are not determined or were deleted for some sale_invoices. And that's why it was left join. And of course its main purpose was to help build incoming payments, that's why it should give information in a situation when multiple incoming payments were not inserted in the base. That's why the Left join appears here. All the rest joins follow foreign keys with the tables that have necessary data, so they all are inner joins. The calculation of 'paid_amount' didn't work as expected, giving us double or triple amounts in testing and raising some unexpected behavior in our following procedure 'generate_payments' until we debugged it and found that the problem stays outside the procedure. We changed the line

```
CAST(SUM(ip.incoming_payment_amount AS DECIMAL(15,2)) AS paid_amount
```

to

```
CAST(SUM(ip.incoming_payment_amount)  /  COUNT(ip.incoming_payment_amount)  AS
DECIMAL(15,2)) AS paid_amount
```

and it worked. We realized that during join the amount of incoming payment appears in each consignment and for each invoice during aggregation it counts up to 3 times, so we should divide SUM to the COUNT to achieve a good result.

f) Creation of the table 'for_debugging_2' that has been created to analyze internal variables in the procedure 'generate_payments' and helped to fix the bug

g) Creation of the procedure 'generate_payments' that:
   i) takes as arguments interval 'days_before' and 'days_after' the date of sale invoice within which payment dates will be randomly generated, and also max_number_of_pmnts_per_invoice
   ii) then it makes cycle and for each invoice in the database it generates from 1 to max_number_of_pmnts_per_invoice (chosen randomly for each invoice) of incoming payments.

The amount of each payment is chosen with randomness within the interval (0, invoice_amount - paid_amount), however with increased probability of full payment of unpaid amount.

h) Execution of procedure ‘generate_payments’ with parameters 10 days_before, 20 days_after and 3 max_number_of_pmnts_per_invoice

i) Execution of the request in sale_invoice_full_aggregated_view to control the result of the ‘generate_payments’ procedure. The result (first rows) on the following picture:

invoice_id	invoice_number	invoice_date	contract_number	contract_date	buyer	buyer_country	invoice_currency	sale_terms	invoice_amount	invoice_amount_CHF	paid_amount
1	INV011	2023-01-11 10:00:00	Contract7	2023-12-05 00:00:00	Sime Darby Plantation Berhad	Malaysia	USD	CIF Gemlik	1211781.33	1119819.24	1120367.54
2	INV012	2023-01-12 11:00:00	Contract21	2023-06-09 00:00:00	Molinos Río de la Plata S.A.	Austria	USD	CIF Tunis	1712627.14	1582655.87	1712627.14
3	INV013	2023-01-13 12:00:00	Contract35	2023-06-26 00:00:00	Pamuk Yem	Turkey	CHF	CIF Malaga	911282.32	911282.32	911282.32
4	INV014	2023-01-14 13:00:00	Contract14	2023-01-27 00:00:00	Marico Limited	Hong Kong	EUR	FOB Odesa	569519.58	567326.93	569519.58
5	INV015	2023-01-15 14:00:00	Contract3	2023-11-10 00:00:00	Cargill, Incorporated	United Kingdom	GBP	CIF Napoli	1708506.77	1929929.25	1708506.77
6	INV016	2023-01-16 15:00:00	Contract28	2023-01-05 00:00:00	Maroc Emirats Arabes Unis Investissement	Morocco	USD	CIF Monopoli	3128830.04	2891383.13	1852316.89
7	INV017	2023-01-17 16:00:00	Contract19	2023-11-18 00:00:00	Agrofert Holding, a.s.	Czech Republic	EUR	CIF Mersin	1048667.44	1044630.08	262166.86

3) ***Hackers_requetes.sql*** : This file contains some SQL requests used for analytics, namely:

a) Creation of ‘buyers_by_product_view’ that represents a nice reusable request of information on sales (calculated total amount in CHF and total volume in measure_unit_short_name) for each buyer and each product. It is a SELECT request from a complex JOIN of many tables, between which only for incoming payment LEFT JOIN applied because for the rest tables we are aware that the correspondent data exist. Grouping has been made up to sale_invoice level to calculate aggregates of our amounts and volumes. Function ‘get_exchange rate’ used to determine exchange rate for conversation to sale_amount_CHF.

buyer	buyer_country	product	sales_amount_CHF	sales_volume	units
AAK AB	Slovakia	Rapeseed meal	2282164.78	10387.680	mt
ADM Milling Company	United Kingdom	Sunflower seed meal pellets	4063552.39	26137.023	mt
Agro-Anadolu Tarım Ürünleri	Turkey	Sunflower seed meal non-pellets	1257853.45	6600.128	mt
Agrofert Holding, a.s.	Czech Republic	Sunflower seed meal pellets	1044630.08	5254.898	mt
Alfa Corporation for Import and Export	Egypt	Crude rapeseed oil	8449765.39	9287.648	mt
Allseeds Black Sea	Ukraine	Sunflower seed meal non-pellets	2255621.04	12552.865	mt
Alwaha for Trade and Marketing Co.	Jordan	Sunflower seed meal non-pellets	2214901.97	10424.674	mt
Archer Daniels Midland Company	United Kingdom	Sunflower seed meal non-pellets	1051210.08	6155.688	mt
Bunge Asia Pte. Ltd.	Czech Republic	Crude sunflower seeds oil	3344894.38	5009.680	mt

b) Creation of ‘sales_statistics_view’ that shows all necessary sales data (by each consignment), including calculated ‘sale_price_CHF’, ‘sales_amount’, ‘sales_amount_CHF’ and other fields from joined tables, to be the main source for our further Tableau visualizations - dashboards ‘Sales by country and buyers’ and ‘Sale prices by country’. It is an atomic (consignment) level join of most of the tables that relate to the sales and don’t relate to payments, that’s why the grouping has not been applied and all joins are inner. CAST(... AS DECIMAL(...)) helps to receive required number format for our calculations and not float with many unnecessary signs after the comma.

sale_date	consignment_id	product	quantity_sold	units	sale_price	currency	sales_price_CHF	sales_amount	sales_amount_CHF	sale_terms	sale_point	sale_point_country	buyer	buyer_country
2023-01-11 10:00:00	2	Rapeseed meal	2598.177	mt	208.92	USD	193.07	542811.14	501617.20	CIF	Gemlik	Turkey	Sime Darby Plantation Berhad	Malaysia
2023-01-11 10:00:00	1	Rapeseed meal	3266.456	mt	204.80	USD	189.26	668970.19	618202.04	CIF	Gemlik	Turkey	Sime Darby Plantation Berhad	Malaysia
2023-01-12 11:00:00	3	Sunflower seed meal pellets	5757.057	mt	196.89	USD	181.95	1133506.95	1047485.11	CIF	Tunis	Tunisia	Molinos Río de la Plata S.A.	Austria
2023-01-12 11:00:00	4	Sunflower seed meal pellets	2880.765	mt	201.03	USD	185.77	579120.19	535170.76	CIF	Tunis	Tunisia	Molinos Río de la Plata S.A.	Austria
2023-01-13 12:00:00	5	Rapeseed meal	4097.124	mt	222.42	CHF	222.42	911282.32	911282.32	CIF	Malaga	Spain	Pamuk Yem	Turkey
2023-01-14 13:00:00	6	Sunflower seed meal non-pellets	2867.816	mt	198.59	EUR	197.83	569519.58	567326.93	FOB	Odesa	Ukraine	Marico Limited	Hong Kong
2023-01-15 14:00:00	8	Rapeseed meal	4631.817	mt	233.05	GBP	263.25	1079444.95	1219341.02	CIF	Napoli	Italy	Cargill, Incorporated	United Kingdom
2023-01-15 14:00:00	7	Rapeseed meal	2728.645	mt	230.54	GBP	260.42	629061.82	710588.23	CIF	Napoli	Italy	Cargill, Incorporated	United Kingdom

c) Creation of ‘sale_invoices_unpaid_view’ that is a filter based on the sale_invoice_full_aggregated_view and forms the base for our Tableau visualization - sheet ‘Unpaid invoices’. It searches only for invoices where paid_amount is less than invoice_amount

invoice_id	invoice_number	invoice_date	contract_number	contract_date	buyer	buyer_country	invoice_currency	sale_terms	invoice_amount	invoice_amount_CHF	paid_amount
1	INV011	2023-01-11 10:00:00	Contract7	2023-12-05 00:00:00	Sime Darby Plantation Berhad	Malaysia	USD	CIF Gemlik	1211781.33	1119819.24	1120367.54
6	INV016	2023-01-16 15:00:00	Contract28	2023-01-05 00:00:00	Maroc Emirats Arabes Unis Investissement	Morocco	USD	CIF Monopoli	3128830.04	2891383.13	1852316.89
7	INV017	2023-01-17 16:00:00	Contract19	2023-11-18 00:00:00	Agrofert Holding, a.s.	Czech Republic	EUR	CIF Mersin	1048667.44	1044630.08	262166.86
9	INV019	2023-01-19 18:00:00	Contract11	2023-10-03 00:00:00	Bunge SA	Czech Republic	GBP	CIF Genova	11991496.53	13545594.48	3330971.26
10	INV020	2023-01-20 19:00:00	Contract6	2023-04-17 00:00:00	Wilmar International Limited	Singapore	USD	CIF Alexandria	9530541.39	8807268.60	2647372.61
11	INV021	2023-01-21 10:00:00	Contract24	2023-08-11 00:00:00	Cairo Oil and Soap Co.	Egypt	CHF	CIF Malaga	2409146.98	2409146.98	502836.82

d) Creation of ‘incoming_payments_view’ - that gathers together all necessary information on the payments to be used further as the base for our Tableau visualization - dashboard ‘Incoming payments’. The Joins are all made INNER as far as we search for information related to existing payments. For them all the related values should just be present. To reuse the data from the Country table twice - once for buyer’s country and once for bank’s country, we made different aliases - ‘buyer_country’ and ‘bank_country’ for different inner joins on different foreign keys relations.

payment_date	payment_amount	currency	payment_amount_CHF	CHF_rate	bank_account	bank_name	bank_country	invoice_number	invoice_date	buyer	buyer_country
2023-01-10 13:00:00	569519.58	EUR	567326.93	0.99615	US0987654321	Bank of America	Unites States	INV014	2023-01-14 13:00:00	Marico Limited	Hong Kong
2023-01-11 12:00:00	911282.32	CHF	911282.32	1.00000	CH1234567890	Swiss Bank Corporation	Switzerland	INV013	2023-01-13 12:00:00	Pamuk Yem	Turkey
2023-01-14 10:00:00	502836.82	CHF	502836.82	1.00000	CH1234567890	Swiss Bank Corporation	Switzerland	INV021	2023-01-21 10:00:00	Cairo Oil and Soap Co.	Egypt
2023-01-15 11:00:00	1712627.14	USD	1582655.87	0.92411	US1234567890	Bank of America	Unites States	INV012	2023-01-12 11:00:00	Molinos Río de la Plata S.A.	Austria
2023-01-16 14:00:00	442346.90	GBP	499675.06	1.12960	GB1234567890	Barclays Bank	United Kingdom	INV025	2023-01-25 14:00:00	Wilmar Europe Holdings B.V.	Netherlands
2023-01-17 11:00:00	1704683.87	USD	1575315.41	0.92411	DE0987654321	Deutsche Bank AG	Germany	INV022	2023-01-22 11:00:00	Golden Agri-Resources Ltd.	Singapore
2023-01-17 14:00:00	1708506.77	GBP	1929929.25	1.12960	GB1234567890	Barclays Bank	United Kingdom	INV015	2023-01-15 14:00:00	Cargill, Incorporated	United Kingdom

e) Creation of ‘average_price_by_product_and_point_of_sale_view’ that takes information from ‘sales_statistic_view’ and calculates weighted monthly average price for each point of sale and product. It will form the base for visualization of ‘Prices’. Dates were transferred by DATE_FORMAT to the months like ‘Jan 2023’. Chat GPT helped to find this function and explain its effect on the data. And for aggregation grouping by product, sale_terms_and_place and sale_month have been applied. ‘Units’ grouping also has been added as far as error raised during testing. In our case all units are the same, so this grouping doesn't affect the results. In other cases, depending on the goal, we could try to find different decisions on that issue. average_price_CHF has been calculated as average weighted by dividing total sum of amounts on total sum of quantities in the rows involved in aggregation.

product	sale_terms_and_place	sale_month	quantity_sold	units	average_price_CHF
Crude rapeseed oil	CIF Malaga, Spain	Jan 2023	2958.368	mt	814.35
Crude rapeseed oil	CIF Seville, Spain	Feb 2023	9287.648	mt	909.79
Crude rapeseed oil	CIF Valencia, Spain	Jan 2023	12302.614	mt	1079.77
Crude rapeseed oil	CIF Valetta, Malta	Feb 2023	3806.564	mt	881.30
Crude rapeseed oil	CIF Valetta, Malta	Jan 2023	13255.316	mt	832.72
Crude sunflower seeds oil	CIF Constanta, Romania	Feb 2023	4830.639	mt	657.63
Crude sunflower seeds oil	CIF Genova, Italy	Jan 2023	5124.177	mt	881.16
Crude sunflower seeds oil	CIF Marseille, France	Feb 2023	11790.635	mt	875.24
Crude sunflower seeds oil	CIF Porto, Portugal	Feb 2023	5009.680	mt	667.69
Rapeseed meal	CIF Alexandria, Egypt	Jan 2023	14420.463	mt	203.58
Rapeseed meal	CIF Gemlik, Turkey	Jan 2023	5864.633	mt	190.94
Rapeseed meal	CIF Malaga, Spain	Jan 2023	4097.124	mt	222.42
Rapeseed meal	CIF Monopoli, Italy	Jan 2023	13549.259	mt	213.40

4. Data Visualizations

We made 5 visualizations - 3 complex dashboards and 2 simple sheets.

1) Dashboard ‘Sales by country and buyers’

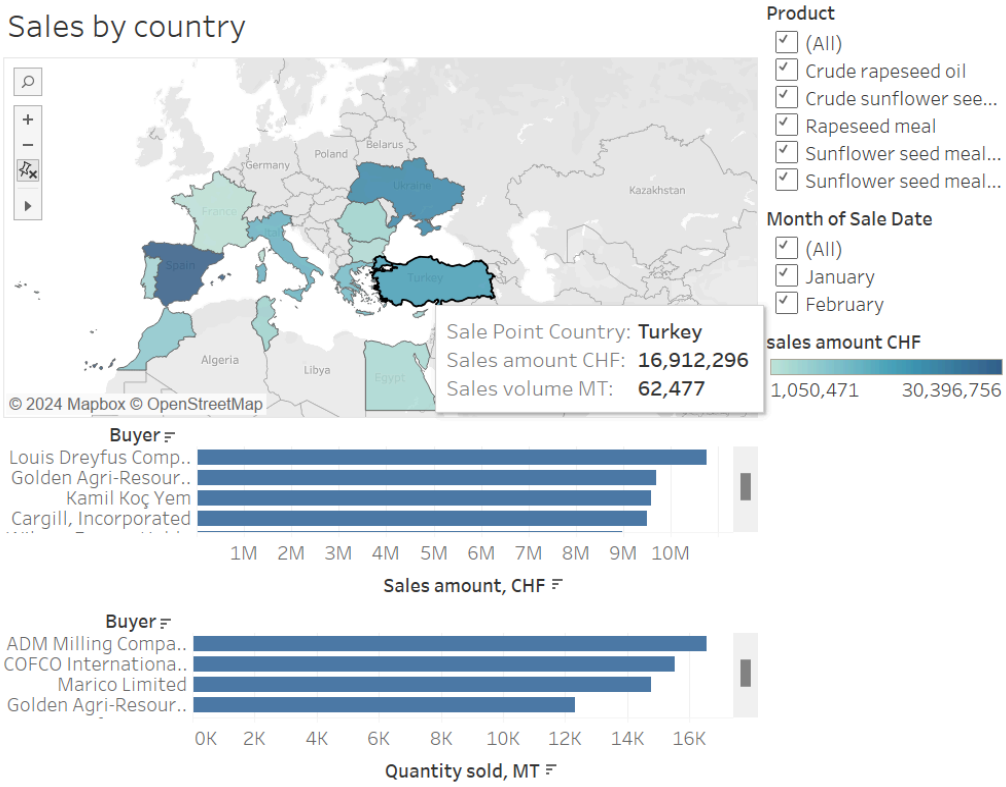
In the first dashboard we have presented a map visualization displaying countries from Europe and a few countries from Africa shaded in varying intensities of blue, representing sales volume. Darker shades indicate higher sales. We also created two horizontal bar charts, where we rank companies based on their total sales amount (converted to CHF) and sales volume (measured in metric tons, MT). We tried to make most of the visualizations interactive so we have interactive filters that provide checkboxes where we can choose products and months (January to February) of our data, we can also see the interval of the total sales amount. Also the filter may be applied by choosing one or more countries on the map. In that case Buyer’s tables will reflect only the shipments with sale_point in those countries.

Process of making this dashboard included few stages:

a) Scripting of ‘sales_statistics_view’ that prepares and calculates all necessary data from database that is related to the sales perspective and will be used in our visualization
b) In Tableau after connecting it through ODBC connector to our local MySQL database (while connecting we had to implement some password to our user ‘root’ because without password Tableau Desktop refused to connect to database), on the ‘Data Source’ tab we had to slide the view to our working space panel, so its data may appear for inserting in Tableau visualizations
c) We made a new sheet ‘Sales_by_country’ where we slided ‘Sale Point Country’ to the blank main field of visualization, where the card appeared. Then we applied SUM(sales amount) to the color button to color the map, added SUM(Quantity Sold) to the Tooltip, applied 2 filters (Product and MONTH(Sale Date)) and performed some minor changes in texting of the tooltips.
d) We made two new sheets ‘Buyer CHF’ and ‘Buyer MT’ where we made horizontal bar graphs representing Buyers sorted by the Sales amount (CHF) and Quantity Sold (MT). They were made by the same scheme

by sliding ‘Buyer’ to the rows and SUM(‘Sales amount CHF’) and correspondingly SUM(‘Quantity Sold’) to the columns and choosing from ‘Show Me’ the right type of the graph. Then, clicking on the top of the graph we sorted the values.

e) We added the previous 3 sheets to the dashboard by sliding and correcting their size. Then we applied all necessary filters, so our dashboard became interactive.



2) Dashboard ‘Sale prices by country and product’

In the second dashboard we implemented the map-based data visualization that displays the average sales prices of various agricultural products across different countries during January and February 2023, together with the pie chart on the tooltip that illustrates the distribution of sales amount by product type. The countries are color-coded based on their average sales price in Swiss Francs (CHF). Through this visualization we can identify which countries have higher or lower average sales prices for these agricultural commodities. The box plot chart shows distribution of sale prices in January and February where the points are colored by each product to give an impression of the price levels of each different product. In the dashboard we also included the filtering by country on map, product and month.

Process of making this dashboard included following stages:

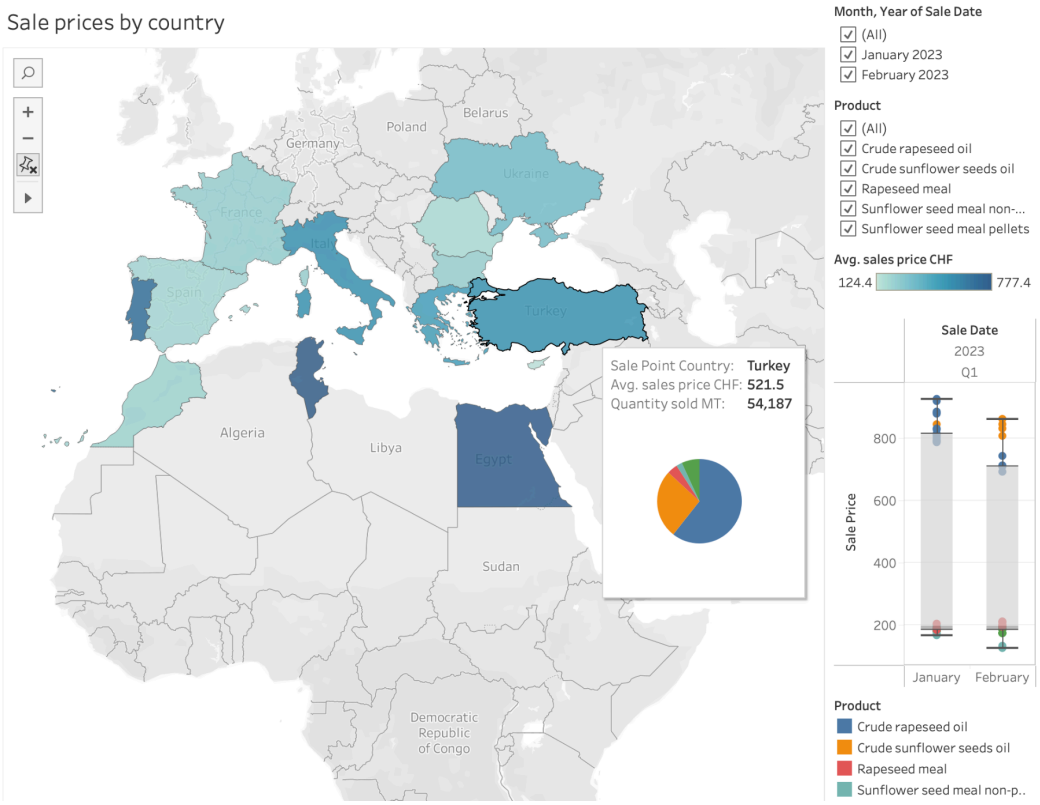
a) It is also based on the ‘sales_statistic_view’, so we didn’t need to perform additional actions to add this view to our canvas in Tableau Desktop

b) We made a simple pie chart ‘Sales by Product’ to show the relative amount of the sold products. We made it by sliding ‘sales amount CHF’ to the rows, ‘Product’ to the color and choosing in ‘ShowMe’ the icon of the pie chart.

c) We made a new sheet ‘Sale prices by country’ where we slided ‘Sale Point Country’ to the blank main field of visualization and the map appeared. Then we colored it by sliding ‘sale price CHF’ to the ‘Color’ button and changing aggregation to ‘AVG’ .Then we added filters by Sale date and product. And also added ‘Quantity Sold’ and pie chart ‘Sales by product’ to the Tooltip.

d) We made a new sheet ‘Distribution of prices’ by sliding ‘Sale price CHF’ to the rows, ‘Sale Date’ to the columns, then choosing boxplot icon in ‘ShowMe’, then adding one more time ‘Sale Date’ to columns and detailed it up to month, changing type of ‘Sale price CHF’ to measure and adding ‘Product’ to the color. Finally we get the colored by product distribution of prices in January and February 2023 with boxplot showing us the Idea of the density of this distribution.

e) On the dashboard we put the sheets ‘Sale prices by country’ and ‘Distribution of prices’ together with the filters by month and product, the legend for product color and also tuned the filters, so by choosing one of the countries on the map and one of the filters by month and product, you may have in the correspondent data to appear in the boxplot view.



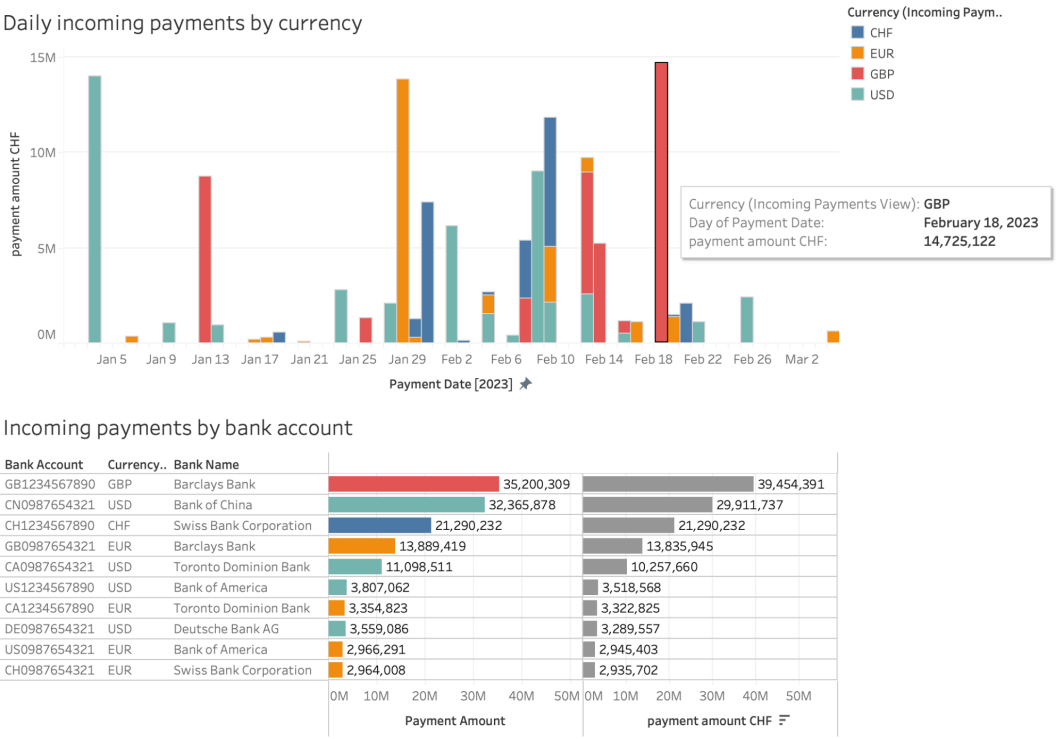
3) Dashboard ‘Incoming payments’

In the third dashboard we may see 2 parts:

The upper part is a vertical stacked bar chart representing the daily incoming payments by currency from January to February. Each currency is represented by its own color, the amounts are converted to CHF to be able to compare them in one chart. On this graph, we can observe daily fluctuations in incoming payments across different currencies. By comparing the heights of bars, we can identify which currency had the highest volume of transactions on specific dates.

The bottom part is a table with horizontal charts representing incoming payments by bank account. Here we can compare payment volumes across different bank accounts both in the currency of the account and converted to CHF for comparison and sorting. By examining the segments within each bar, we can see which

accounts received the most incoming payments and in which currency. By choosing one or more bank accounts from the bottom part we filter the data shown in the upper part.



Process of making this dashboard included following stages:

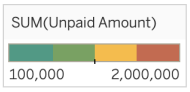
a) This dashboard is based on the ‘incoming_payments_view’, so first of all this view has been added to our database
b) We added this view to Tableau by sliding it to canvas and correcting relations, as far as without correction Tableau didn’t want to show the data. As far, as we didn’t use this view together with other views in the same visualization, for us the nature of connection wasn’t important and we connected the first fields that we found similar in different views.
c) We made a new sheet ‘Daily payments by currency’ by sliding ‘payment amount CHF’ to the rows, ‘payment date’ to the columns, ‘Currency’ to the color. Then we chose from ‘ShowMe’ the stacked bar chart icon and corrected the date format to be ‘Day’. Then we fixed the range of the x axis, so it doesn’t change during filtering for better visibility and comparison.
d) We made a new sheet ‘Incoming payments by bank account’ by sliding ‘Bank Account’, ‘Currency’ and ‘Bank Name’ to the rows and ‘Payment Amount’, ‘Payment Amount CHF’ to the columns. Then we slided ‘Currency’ to the color of the first column, slided ‘Payment amount’ to the label of the first column and ‘Payment amount CHF’ to the label of the second column. Finally we sorted the second column by clicking on top of it.
e) We added both sheets to the dashboard and arranged filtering of upper part by the choice in bottom part

4) Visualization sheet ‘Unpaid invoices’

This visualization has as its goal to show to the finance department the detailed list of unpaid invoices. It is grouped by currency and sorted by amount. The color scheme is chosen to highlight the importance of the higher unpaid amounts compared to the lower ones. Totals by each currency applied.

Unpaid invoices

Invoice Currency	Invoice Number	Buyer (Sale Invoices Unpaid View)	Invoice Amount	Paid Amount	Unpaid Amount
CHF	INV044	Bunge Asia Pte. Ltd.	19,628,035	4,907,009	14,721,027
	INV031	Südzucker AG	749,981	125,887	624,094
	Total		20,378,016	5,032,895	15,345,121
EUR	INV027	Hacı Hasan Yağ Sanayi	2,340,484	381,190	1,959,294
	INV014	Marico Limited	2,136,252	281,965	1,854,287
	INV046	Louis Dreyfus Company B.V.	2,291,130	636,425	1,654,705
	INV017	Agrofert Holding, a.s.	1,967,978	546,661	1,421,317
	Total		8,735,844	1,846,240	6,889,604
GBP	INV047	Alwaha for Trade and Marketing Co.	23,363,624	6,489,896	16,873,729
	INV025	Wilmar Europe Holdings B.V.	1,086,768	776,526	310,243
	Total		24,450,393	7,266,421	17,183,971
USD	INV020	Wilmar International Limited	5,395,587	1,348,897	4,046,690
	INV012	Molinos Río de la Plata S.A.	2,586,249	1,041,520	1,544,730
	INV035	COFCO International Ltd.	1,881,947	1,244,735	637,212
	Total		9,863,783	3,635,151	6,228,632



Process of making this visualization included following stages:

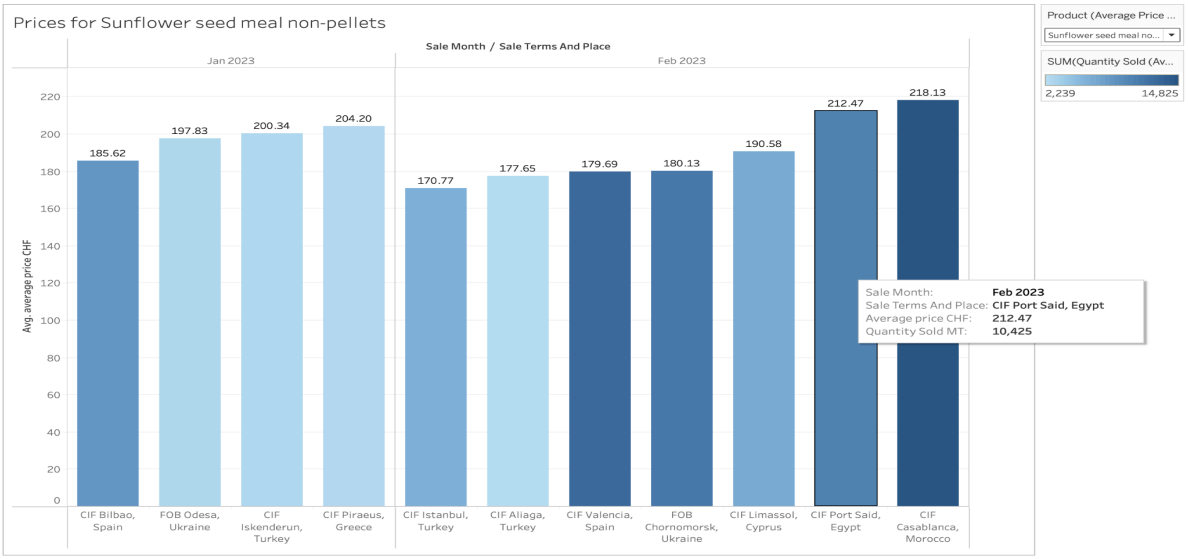
a) It is based on the 'sale_invoices_unpaid_view', so we firstly added this view to our database

b) We added this view to Tableau Desktop on 'Data Source' tab by sliding to canvas and correcting relation (with relation - same approach as above - because for our goal it isn't important)

c) We made a new sheet and slided 'Invoice Currency', 'Buyer', 'Invoice number' to the rows, then 'Invoice Amount', 'Paid amount' and 'Unpaid amount' to the columns, then chosen in 'ShowMe' the text table icon. Then we added 'Unpaid amount' to the color, then changed the color scheme to 4 steps from green to red. Then we sorted by 'unpaid amount' and added totals by each currency.

5) Visualization sheet 'Prices'

This visualization has as its goal to show the average monthly sale price for each product by each point of sale and sale terms. Visualization is interactive, you choose the product from the dropdown filter menu and obtain the graph for this product. Each bar is colored by shades of blue - the more dense the color - the more the volume of sales. Tooltip shows all necessary data involved for each bar.



Process of making this visualization included following stages:

a) It is based on the 'average_price_by_product_and_point_of_sale_view', so we firstly added this view to our database
b) We added this view to Tableau Desktop on 'Data Source' tab by sliding to canvas and correcting relation (with relation - same approach as above - because for our goal it isn't important)
c) We made a new sheet and slided 'average price CHF' to the rows, then 'Sale Month', and 'Sale terms and place' to the columns, then corrected date format representation. Then we added 'Quantity sold' to the color. Then we added a filter by product, showed it and changed its type to a dropdown menu with 1 choice. Then we added necessary data into the tooltip and corrected the appearance, so the text in the bottom could be more readable.

5. AI Model Usage

Chat GPT revealed itself as a reliable co-pilot by supporting us during this project and significantly reducing the time of development. We used it for the following purposes:

1) To generate some reliable manually inserted data. We gave it the structure of our table written in SQL and asked to provide data with certain parameters (for example buyers should be famous company names in the agriculture sector). After some small adjustments, we received SQL code fragments for manual insertion of this generated data.
2) While writing our SQL code we consulted with Chat GPT for the names and parameters of certain SQL in-built functions and syntaxis.
3) While testing our code, we gave him our code and he mentioned possible problems with it. Not alway he found the problem, but his replies hinted us where to search for the problem and the problem solution has been found much quicker.

So, using ChatGPT, we realized that it cannot do all the work for you, but it may significantly improve productivity and save time during all the phases of the project.

6. Work Distribution

In our group, we valued teamwork and collaboration. We started by sharing ideas and collectively selecting the best one. We used Trello board to define our tasks and put them into cards. Then, each one took the task that he will make and worked with it. Sometimes we worked in pairs that showed a good improvement in productivity. Regular meetings allowed us to track progress and address any challenges together. Attending class regularly was crucial for us to seek feedback and enhance our project. We utilized GitHub for better version control and merge of our independent work. In conclusion we had a task based distribution of the work, with multiple sprints. We obtained a good experience of modern methods of collaboration that are used throughout the industry.

7. Answer for 'Bonus' Question

In the scenario where our enterprise with our MySQL base will merge with the other that has its own NoSQL solution, we'd perform the following analysis on how to allow our databases to work together in a new formed commercial unit:

- If the goal of both bases is to visualize the sales of each unit, we could leave our solutions separate as long as Tableau can also import data from their database. Thus we could merge the data for our visualizations directly in Tableau. Only some conventions can be applied on certain fields to simplify the stage of visualization.

- If the goal is to develop a more unified approach for collection, storage and utilization of the data, we should apply the steps ETL (Extract, Transform and Load) to our data to bring it to the newly formed Data Warehouse. The data from our database could be denormalized in the same way that we made it in this project by generating some views that help analytic systems to perform faster on this set of data.
- The NoSQL decision used in a new enterprise could be document (JSON) based like Mongo DB. This is a rather common standard for storing various data that has no fixed structure. There are a lot of advantages to perform this approach, starting with the presence of a lot of qualified specialists in MongoDB and ending with the presence of many integration solutions for this type of NoSQL base.
- Tableau understands JSON format and join of the two bases will take only one step - by adding a new level of key-value for separation of the data of our database, other unit database and newly common database. It will only add one more feature to Tableau data source that we may use in the same reports by filtering on this feature.

8. Conclusion

The project "ERP Sales of Sunolta" provides a useful solution for visualization of sale processes in modern international wholesale companies. It can and probably will be reutilized to develop a more complex system integrated in the whole Data Warehouse strategy of such a company or even group of companies.

All the code and visualizations can be applied to the real company data and will bring a positive impact on the transparency of the sales processes for the company's management and lead to a more informed taking of management decisions. Work of this project thus serves as a practical application of academic concepts to real-world business challenges, emphasizing the importance of integrated data systems in modern enterprise management.

During the process of development the main problems to solve were the complexity of ERP system and impossibility to implement all of its features in this project. So we took an agile approach to make something 'small', but 'working' and bringing some necessary utility that can be improved and integrated in the future more robust system.

One more class of problems is the specifics of the MySQL system that does not accept some SQL code, for example, we figured out that none of the code except declaration of variables may be inserted in the procedure before the declaration of another variable. It is impossible to declare variable a, then put some complex initial value to it using 'SET' or 'SELECT INTO' and then declare variable b. It will raise an error and from the error message it is not understood why.

The easiest part of the project was to create database structure after we approved our relational model.

9. References

1) https://github.com/timkesh/Base-de-donnees-ERP_Sales.git - Our project files
2) Modélisation des bases de données : [UML et les modèles entité-association]
   Brouard, Frédéric (auteur_e); Soutou, Christian (créateur_trice), Paris : Eyrolles
   4e édition; [2017]
3) https://www.snb.ch/en/the-snb/mandates-goals/statistics/statistics-pub/current_interest_exchange_rates#t000 - source of the currency exchange rates
4) Production – Sunolta - wholesale company corporate site - source of our inspiration
5) https://trello.com - Trello application for project management used in our work