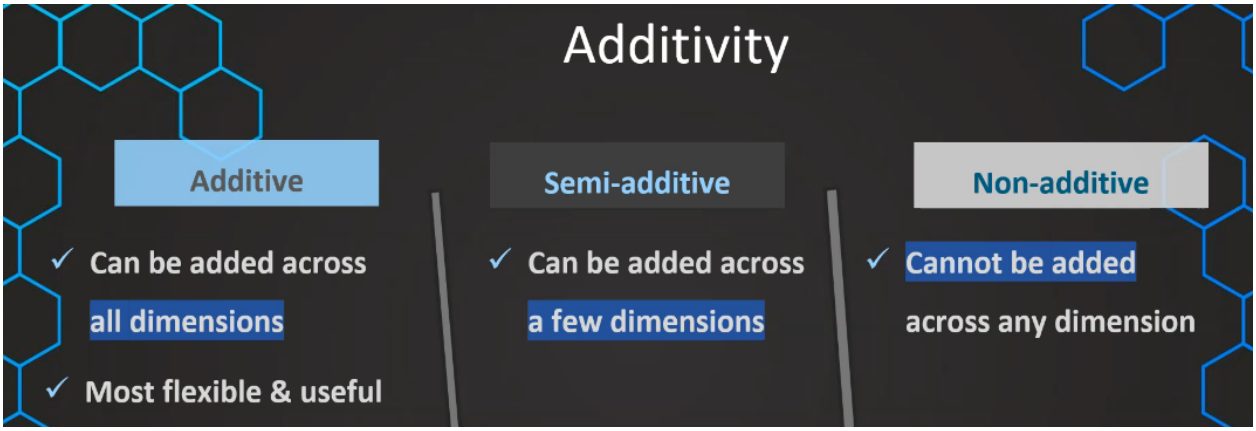
06. Facts

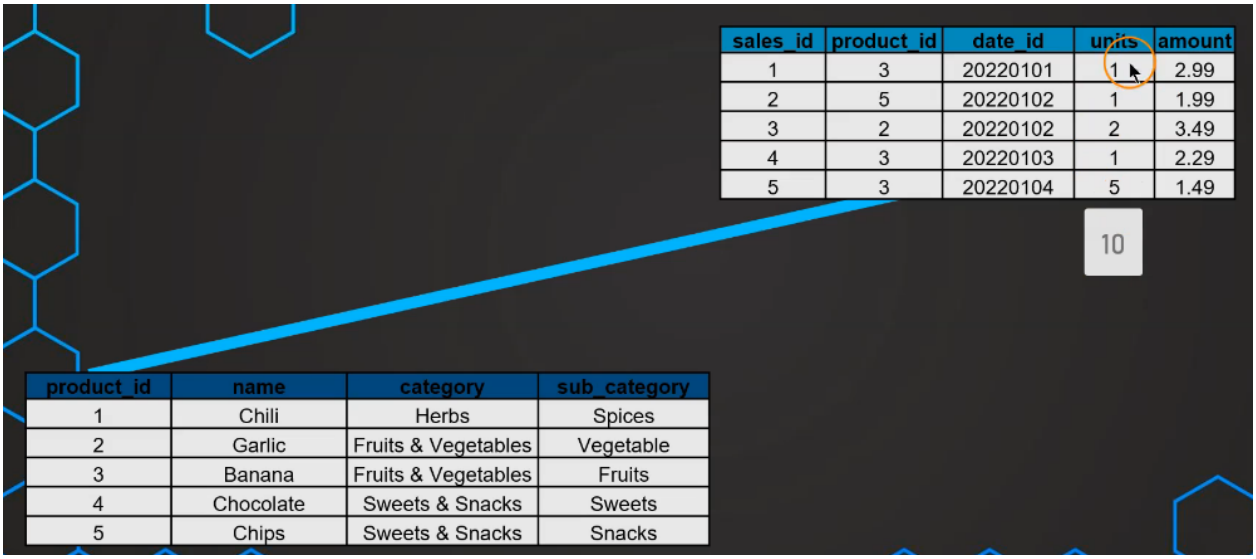
Additivity

- Usually facts are numerical values and numerical values, it's very intuitive to add those values up.
- And this is what we usually want to do in our analytics in Power BI reporting to add those values up to totals.
- But now this cannot be done for all of the facts because the totals do not always make sense. And therefore we want to distinguish between three different types of additivity in our facts.

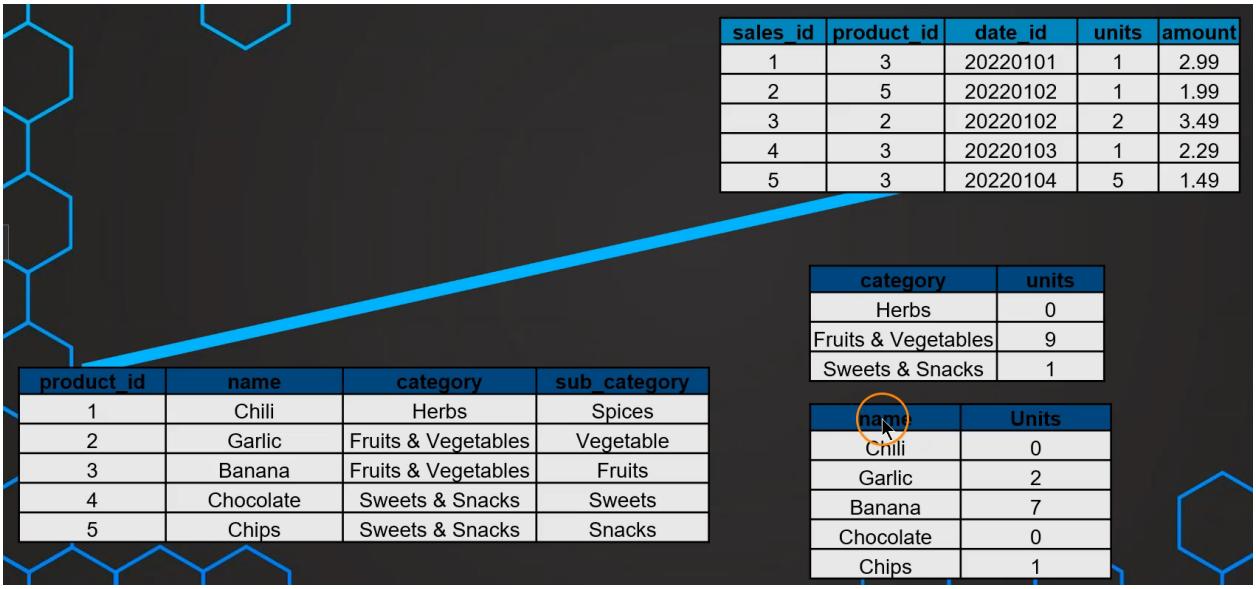


- The first one is the fully-additive fact. This is the most common type, and it can be just normally added up across all of our dimensions.
- And the result, the total, of course still is meaningful and makes sense and this is the most flexible and most useful fact because it can just be added up across all of the dimensions and has the most analytical value.
- But then we also have the semi-additive fact which can only be added up across a few dimensions. So it's again, less flexible.
- Then last but not least we also have the completely non-additive fact. This can not be added up whatsoever across any dimension and we will also have a look at how we can deal with these non-additive and semi-additive facts.

- But first, let's understand the common type the additive fact. In this case, we see that we have a sales table of five different sales and we have the fact of the units sold.



- So for every of the sales, for every row we see for example, sale number three, we have two units sold and now it makes sense to add up all of these values. In total, we have sold 10 units.
- This is a number that makes sense that is meaningful and we have gained that number by just adding up all of these values.
- So across all of the dimensions we can add up these values and the number makes sense.
- So for example, we can group by the category and get meaningful numbers that make sense. And also we can just group by, for example the name and the numbers to make sense.



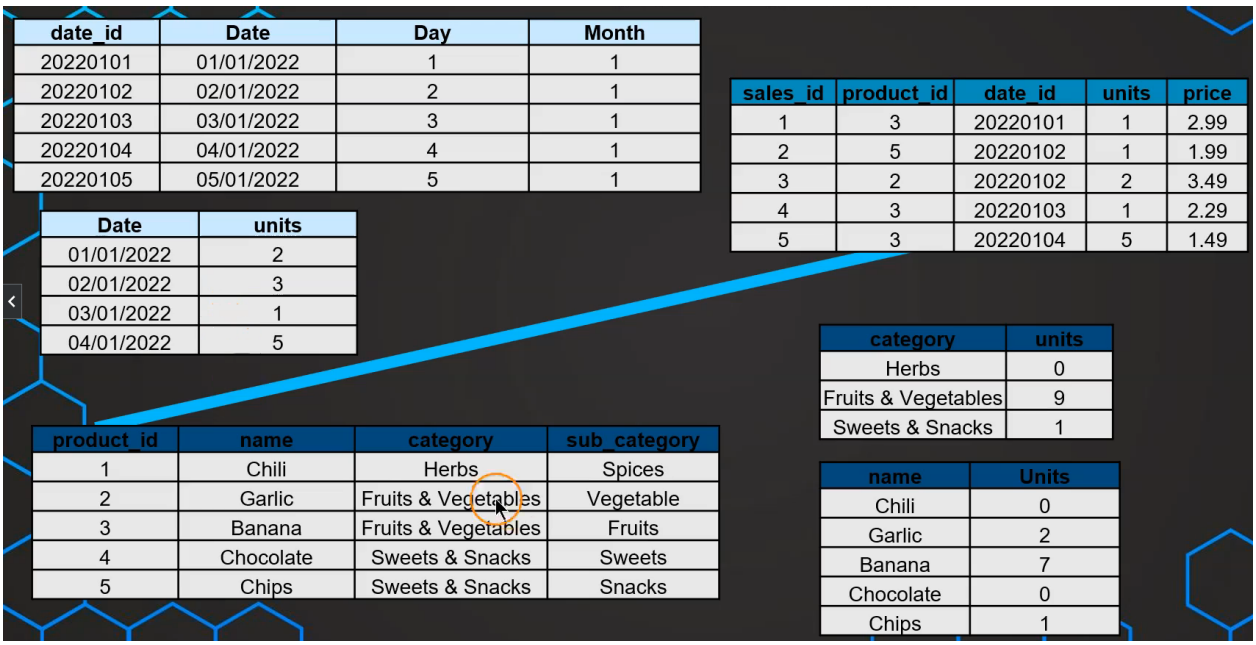
sales_id	product_id	date_id	units	amount
1	3	20220101	1	2.99
2	5	20220102	1	1.99
3	2	20220102	2	3.49
4	3	20220103	1	2.29
5	3	20220104	5	1.49

category	units
Herbs	0
Fruits & Vegetables	9
Sweets & Snacks	1

product_id	name	category	sub_category
1	Chili	Herbs	Spices
2	Garlic	Fruits & Vegetables	Vegetable
3	Banana	Fruits & Vegetables	Fruits
4	Chocolate	Sweets & Snacks	Sweets
5	Chips	Sweets & Snacks	Snacks

name	Units
Chili	0
Garlic	2
Banana	7
Chocolate	0
Chips	1

- And also this is the case, if we want to see this across the dates, we can add the values up across all of the categories across all of the product names. And then for every single day we get the number of units sold. And of course, the number is meaningful and make sense.



date_id	Date	Day	Month
20220101	01/01/2022	1	1
20220102	02/01/2022	2	1
20220103	03/01/2022	3	1
20220104	04/01/2022	4	1
20220105	05/01/2022	5	1

sales_id	product_id	date_id	units	price
1	3	20220101	1	2.99
2	5	20220102	1	1.99
3	2	20220102	2	3.49
4	3	20220103	1	2.29
5	3	20220104	5	1.49

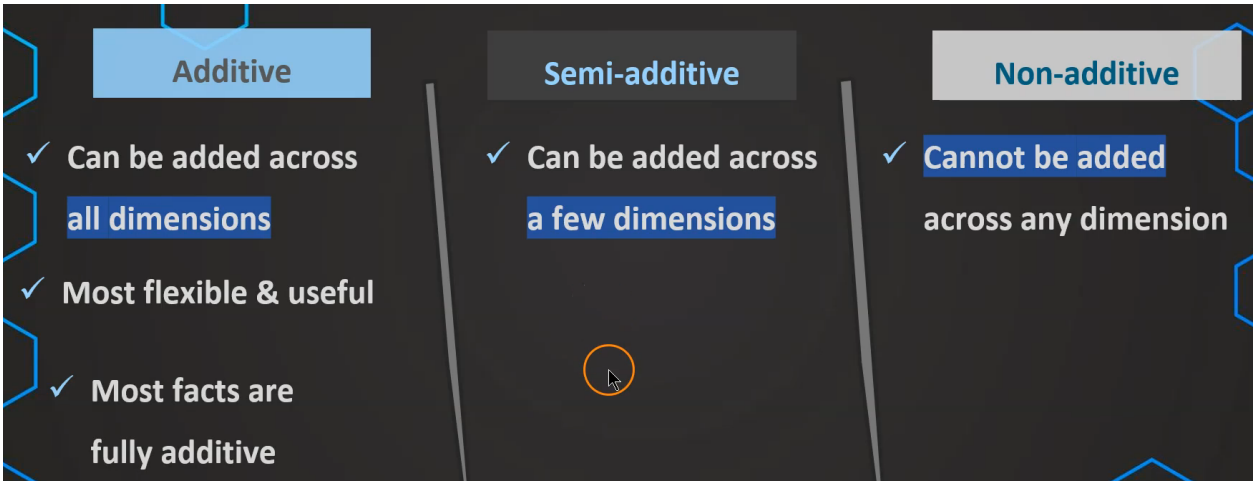
Date	units
01/01/2022	2
02/01/2022	3
03/01/2022	1
04/01/2022	5

category	units
Herbs	0
Fruits & Vegetables	9
Sweets & Snacks	1

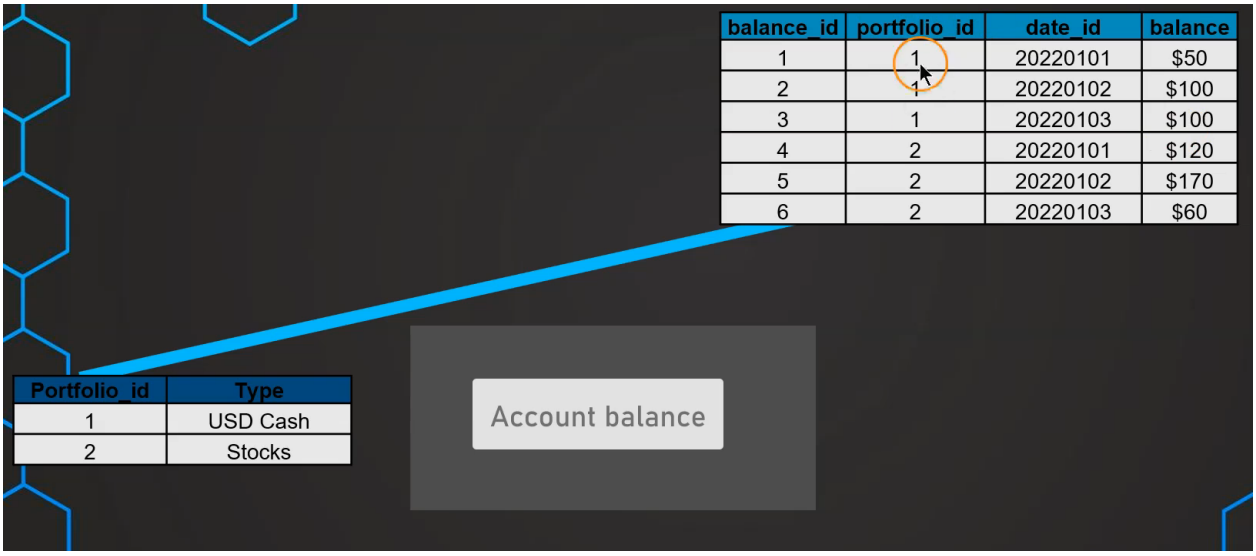
product_id	name	category	sub_category
1	Chili	Herbs	Spices
2	Garlic	Fruits & Vegetables	Vegetable
3	Banana	Fruits & Vegetables	Fruits
4	Chocolate	Sweets & Snacks	Sweets
5	Chips	Sweets & Snacks	Snacks

name	Units
Chili	0
Garlic	2
Banana	7
Chocolate	0
Chips	1

- So this is the most common type of our facts

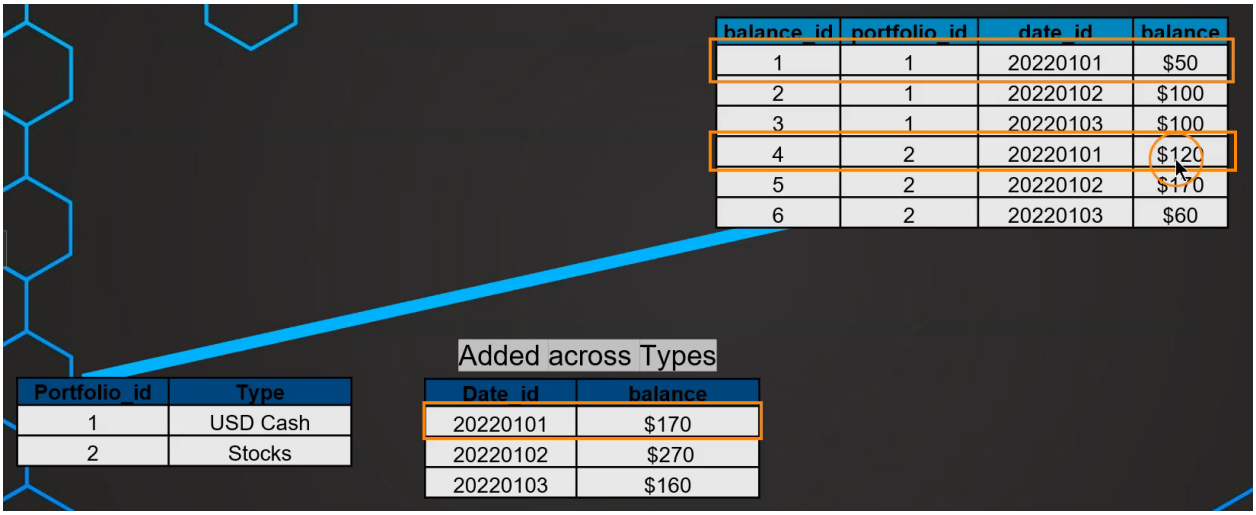


- But now we also have the semi-additive fact and now we also want to see an example for that.
- So let's assume, and this is the most common example for a semi-additive fact, we have account balances.

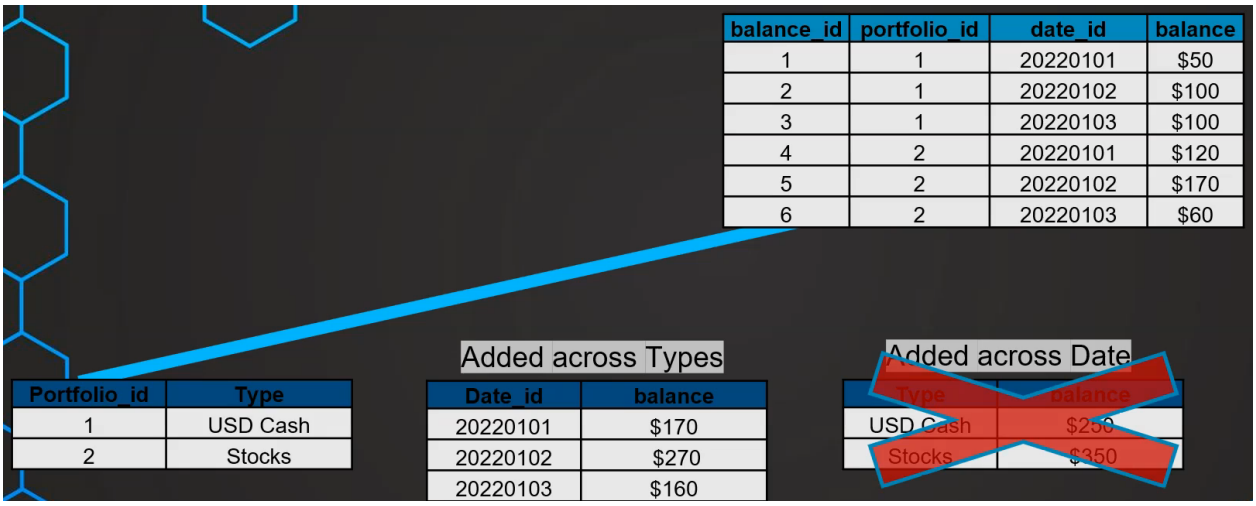


- In this case, we have one row where we have in the first day an account balance of $50 and now we add $50 to that account. So in the next day, the account balance is now $100. And then again, in the next day, there is no change. We have no money coming in and no money going out and therefore the value is still 100.

- But now of course, it doesn't make sense to add up all of these values and get a total balance because the number in the end we have on our account is just $100 on the portfolio type one, of course.
- But now what we can do is we can indeed add the values up across the portfolio types. So for example, for day number one, so we don't add up across the dates, but only across the portfolio type. So we have $50 plus in the same day portfolio type two, $120. So 50 plus 120 is 170. And then in the second day we have in total $270.



- So this does make sense if we want to add up those values across the portfolio types but now we cannot add them up across the dates.
- So if we want to add up the total amount of balance this doesn't really make sense because in fact, in the end we only have $100 in USD cash and in the last day we have only $60 in portfolio type two.



balance_id	portfolio_id	date_id	balance
1	1	20220101	$50
2	1	20220102	$100
3	1	20220103	$100
4	2	20220101	$120
5	2	20220102	$170
6	2	20220103	$60

Portfolio_id	Type
1	USD Cash
2	Stocks

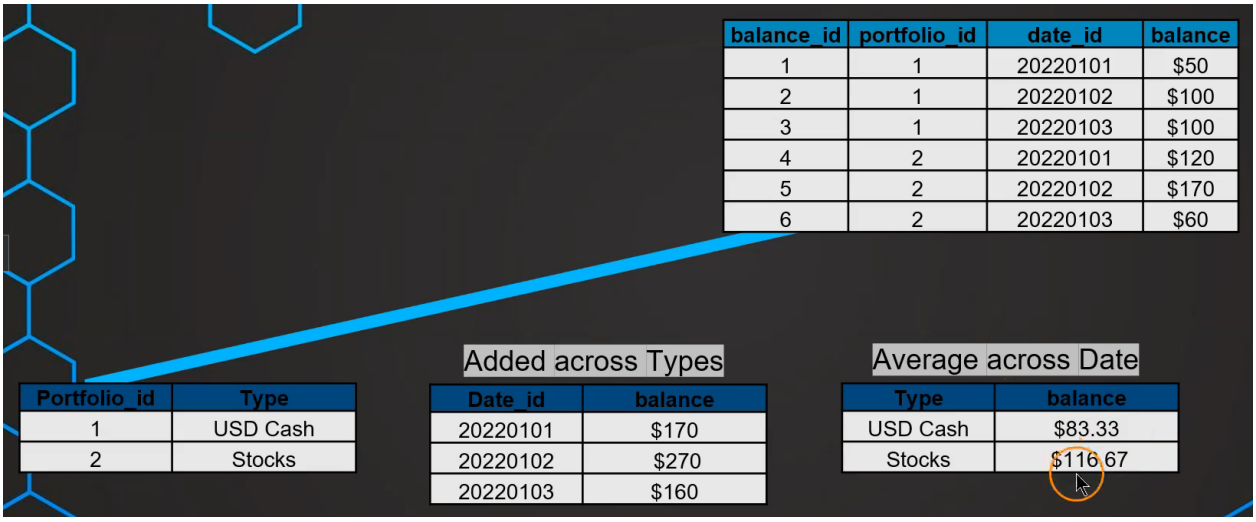
Added across Types

Date_id	balance
20220101	$170
20220102	$270
20220103	$160

Added across Date

Type	balance
USD Cash	$250
Stocks	$350

- So the stocks, and therefore this is not possible to add up these values across the date because the numbers simply don't make any sense. And therefore we cannot do that.
- And the date is also the typical example of a dimension that the values in a semi-additive fact cannot be added up across.
- So this is a very typical example here, but an alternative could be to just take the average.



balance_id	portfolio_id	date_id	balance
1	1	20220101	$50
2	1	20220102	$100
3	1	20220103	$100
4	2	20220101	$120
5	2	20220102	$170
6	2	20220103	$60

Portfolio_id	Type
1	USD Cash
2	Stocks

Added across Types

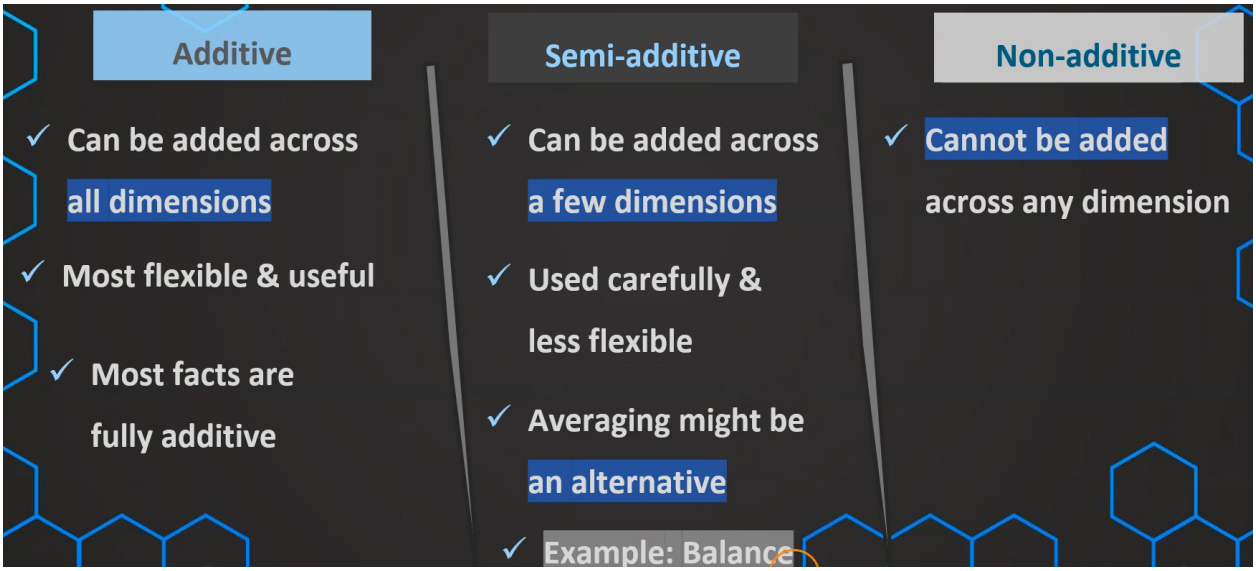
Date_id	balance
20220101	$170
20220102	$270
20220103	$160

Average across Date

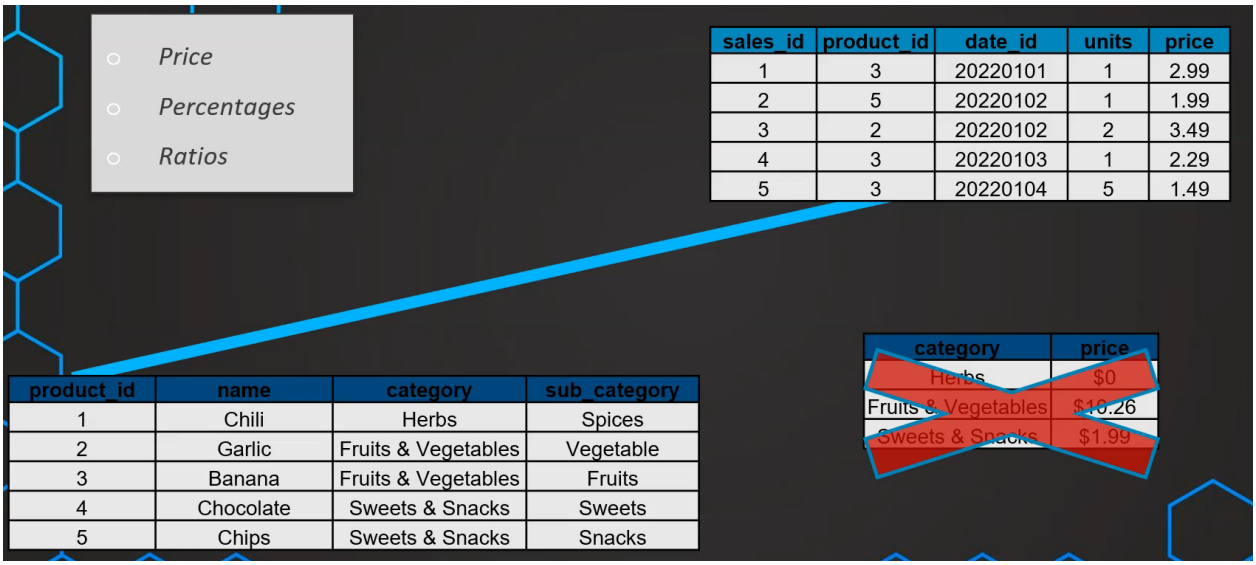
Type	balance
USD Cash	$83.33
Stocks	$116.67

- Again, this could be across the date, and in this case the average could make sense. So we can have a daily average and then get these numbers that we see.

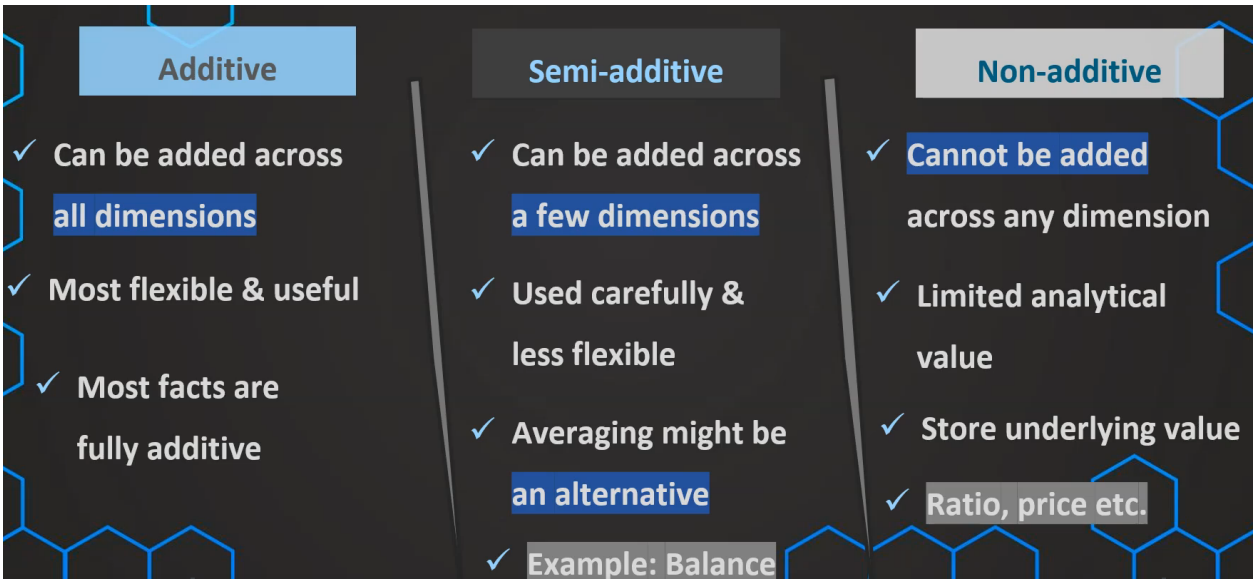
- So this could be something that is possible for the semi-additive facts, but in general, we have to be very careful to use them in the right way so that we are not adding them up across all of our dimensions.
- So this is a typical example that we've seen the account balance.
- So this is not possible to add up across the data dimension



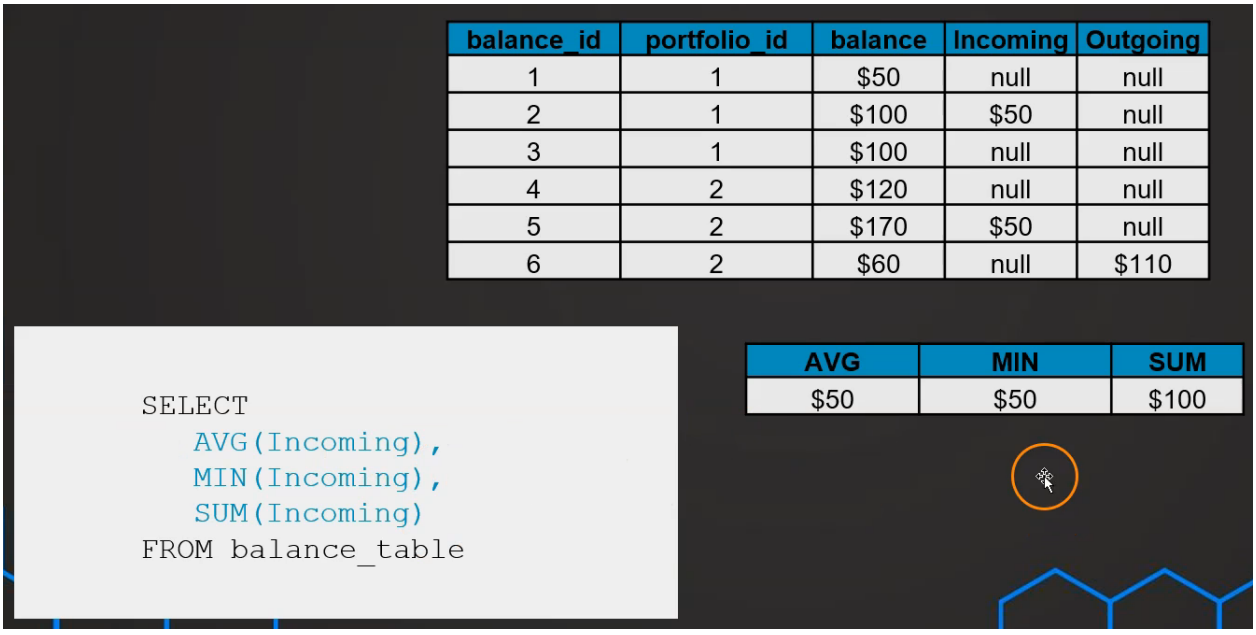
- But now how does it look like if we have a non-additive fact? A non-additive fact is something that we have seen in here where we have the price of a unit and now that we can have multiple units sold, and then we if we want to have the total revenue would have to multiply the unit sold with the price per unit.
- And of course, the revenue is fully additive but the price itself is not additive because we cannot add up all of the prices and then get something like a total price of a category, for example. This doesn't make any sense, and therefore this is completely non-additive because we cannot add those values up in any way and get a meaningful value.



- And therefore, in this case, we have a non-additive fact.
- And even the average is very tricky and we have to still consider the number of units sold to get a weighted average.
- So with these non-additive facts, we have to be very careful. And typical examples for those are, as we've seen the price, but also oftentimes some percentages. So we cannot add them up usually.
- And then also ratios. So again, ratios cannot be added up. So for example what is the inventory level in our warehouse? These are usually ratios that don't make sense to be added up.
- And now how do we deal with those non-additive facts? Because they have limited analytical value, some even argue to include them in a fact table.
- A better method would be to store the underlying value. So for example, if we have some ratio we can store both the numerator and the denominator and then calculate our resulting values in Power BI tools.
- So for example, in Power BI, we can make our own calculations with those underlying values. And with that we have the best analytical value from these non-additive facts.
- So those non-additive facts have the lowest analytical value and we have to be most careful with them. And therefore it is good to know about these three different types of facts in terms of the additivity.



Nulls in facts



balance_id	portfolio_id	balance	Incoming	Outgoing
1	1	$50	null	null
2	1	$100	$50	null
3	1	$100	null	null
4	2	$120	null	null
5	2	$170	$50	null
6	2	$60	null	$110

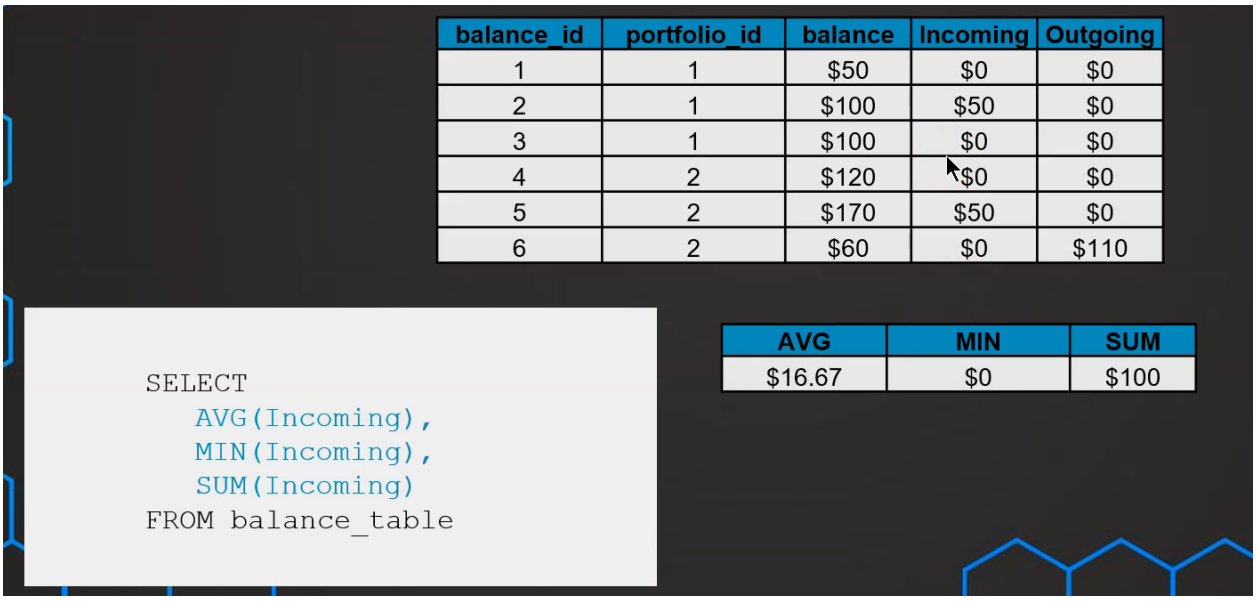
```
SELECT
   AVG(Incoming),
   MIN(Incoming),
   SUM(Incoming)
FROM balance_table
```

AVG	MIN	SUM
$50	$50	$100

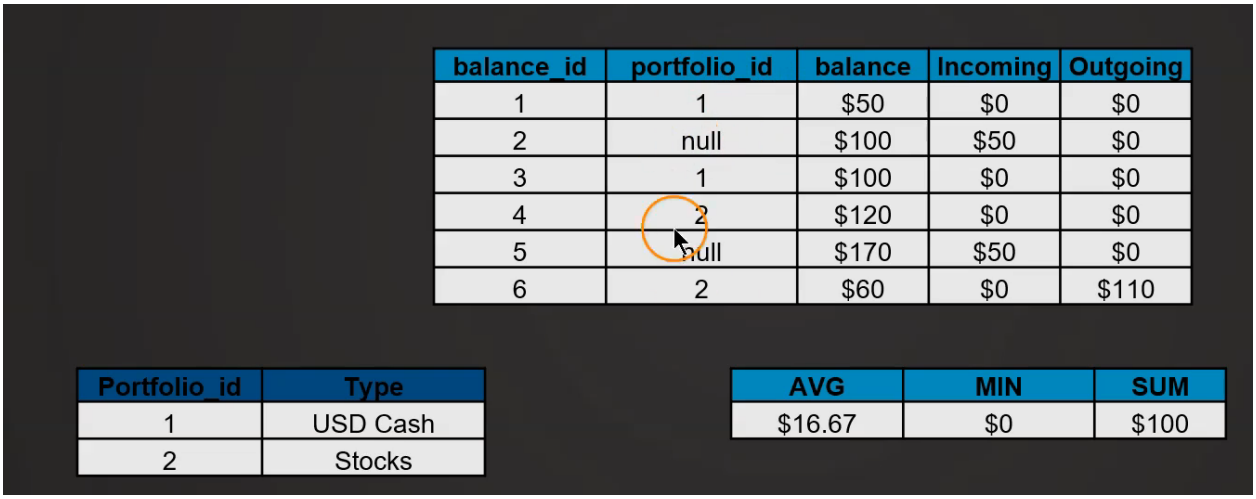
- Now that we've already talked about the additivity, we also want to talk about what happens if we have nulls in our facts, which can be sometimes the case.
- So therefore, let's have a look at how we can deal with that.
- Usually, if we have nulls, this is not a problem at all because all of the aggregations that we are doing in SQL are also in our BI tools, such as Power

BI, Tableau, and so on.

- These null values can be dealt with very easily.
- So for example, in SQL, if we want to take the average of the incoming amount in our account, so for example, we have $50, the next day we have 100, so there were $50 incoming and then the next day, there was no change, so nothing was incoming and therefore, we have a null value in here.
- But now if we want to aggregate that table, so all of the average, minimum, sums, we can easily get those numbers.
- So this is something that SQL and all of the other tools usually can deal with very easily. So therefore, this is usually nothing that we need to take special care about.
- But now oftentimes, the only thing that we need to be a little bit careful is that if we, for example, want to have a look at the average, we see the average incoming amount is indeed 50 US dollars because when there were some transfers, the average of these transfers were $50.
- But the meaning can be a little bit misleading because on average, we did not get 50 US dollars per day, for example. And therefore, in some cases, it can make sense to replace the null values with real zeros if the meaning should be there was zero but also null values can occur.



- But then if they are real null values, this is completely fine because all of the aggregations can still be done and make sense.
- The only thing that we need to be careful with nulls is in the foreign keys.

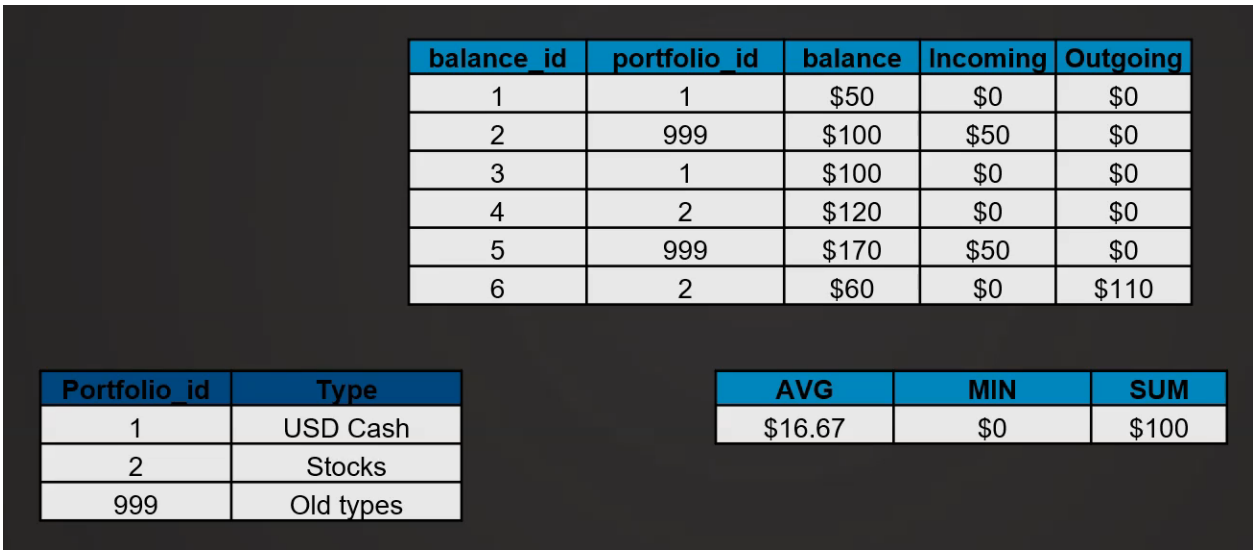


balance_id	portfolio_id	balance	Incoming	Outgoing
1	1	$50	$0	$0
2	null	$100	$50	$0
3	1	$100	$0	$0
4	2	$120	$0	$0
5	null	$170	$50	$0
6	2	$60	$0	$110

Portfolio_id	Type
1	USD Cash
2	Stocks

AVG	MIN	SUM
$16.67	$0	$100

- So if we have a fact table and then there is, for example, no portfolio ID associated, we should not use values here at all because this can create some conflicts and lead to some missing data and some problems if we want to connect this to some dimension tables.
- In this case, if we have null values because there is no portfolio type available, then we should even create dummy values.



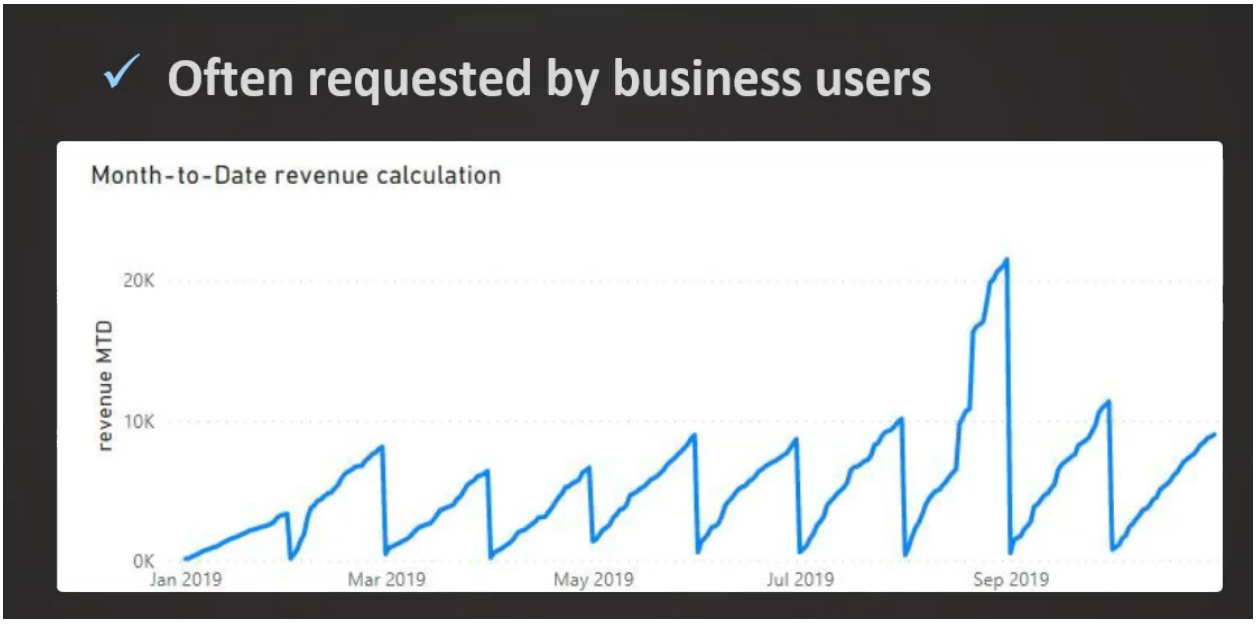
balance_id	portfolio_id	balance	Incoming	Outgoing
1	1	$50	$0	$0
2	999	$100	$50	$0
3	1	$100	$0	$0
4	2	$120	$0	$0
5	999	$170	$50	$0
6	2	$60	$0	$110

Portfolio_id	Type
1	USD Cash
2	Stocks
999	Old types

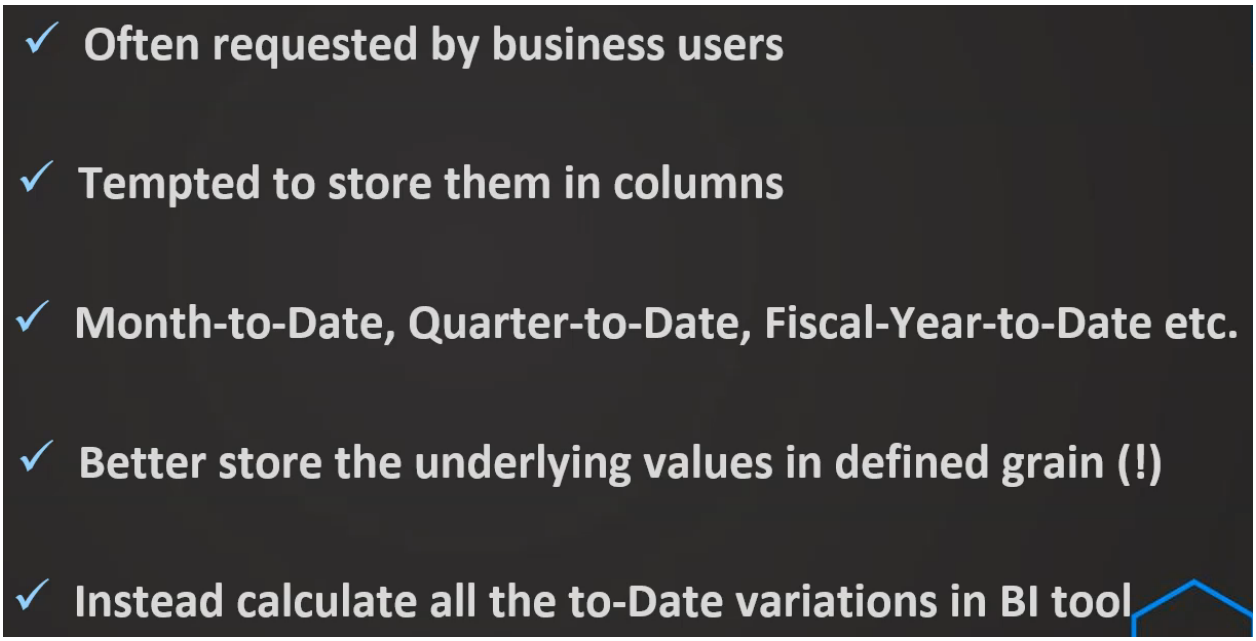
AVG	MIN	SUM
$16.67	$0	$100

- So for example, 999 or also any other number, for example, minus one or something that is quickly indicating that this is a special value. And then we can also add this value of 999 into our dimension table.
- So for example, it could be outdated accounts or sometimes also, we have no date associated. And then we can also just include some dummy value and include some dummy date, for example, 1st of January 1900. And then we have all of the data and we can easily still connect those tables with the dimension and make sense of the data and don't lose any data.
- But in general, we can easily deal with these null values and it's completely fine if we have in our facts null values. But there's one type of fact that we should not include in a fact table.

Year-to-date facts



- So these are specific calculations that are actually very problematic, and therefore, we want to discuss how we can deal with those year-to-date calculations in our data warehouse.



- So this is something that is very commonly requested by the business users because they want to have those year-to-date or month-to-date calculations in their BI tool because they want to see it in their reportings.
- And therefore, we are tempted to just implement this by calculating those year-to-date or month-to-date facts and physically store them in columns in our data warehouse. And this can be true also for the month-to-date calculation, quarter-to-date, fiscal-year-to-date and all of the different variations of basically, for example, just the revenue. So just the underlying value.
- And we have all of these different variations. And this is now actually very problematic because these calculations are not in the defined grain of our fact table because you remember, our fact table always has a defined grain.
- For example, the daily level. For example, if we want to store the revenue, then one row contains just the value of the revenue for each and every day. And now if we have values that are not in this same grain, this is very problematic if now the users, the end users want to aggregate those values.
- So we want to make random aggregations across different dimensions across the date dimension and so on. And then, of course, this will lead to wrong calculations and wrong aggregations.

- The values will just be overstated and in fact, just not true. And therefore, this is something that we should avoid.
- So we should not store those results physically in our data warehouse. The better alternative is to just store the underlaying value, for example, just the revenue in the defined grain, for example, in a daily grain.
- And then, of course, we need to fulfill the need of the business users, and we can do that in the way of just calculating those to-date variations.
- This is possible in all of the BI tools such as Power BI, Tableau, and also, OLAP cubes can handle those year-to-date or month-to-date calculations very well and this is now the preferred method.
- So we should not store the calculations physically in our data warehouse, but just calculate them in our BI tools.

Quiz

Good job!
Yes we can aggregate the items to see how many items are in stock in total. But we cannot aggregate them across the date dimension.

Question 1:

This table displays the number of items that are in stock in the company's warehouse for each product in each day.

Date	Product_FK	Items in stock
20-07-2022	1	15
20-07-2022	2	1
20-07-2022	3	18
20-07-2022	4	6
21-07-2022	1	34
21-07-2022	2	14
21-07-2022	3	34
21-07-2022	4	10
22-07-2022	1	27
22-07-2022	2	14
22-07-2022	3	31
22-07-2022	4	12
23-07-2022	1	13
23-07-2022	2	4
23-07-2022	3	15
23-07-2022	4	8

Is the fact "Items in stock" additive, semi-additive or non-additive?

- ○ Non-additive
- ◉ **Semi-additive**
- ○ Additive

Good job!
We can add up the products sold across all dimensions and we get a number that makes sense - total number of items sold.

Question 2:

This table displays the number of items that have been sold by the company for each product in each day.

Date	Product_FK	Quantity sold
20-07-2022	1	15
20-07-2022	2	24
20-07-2022	3	0
20-07-2022	4	19
21-07-2022	1	8
21-07-2022	2	21
21-07-2022	3	22
21-07-2022	4	17
22-07-2022	1	6
22-07-2022	2	32
22-07-2022	3	24
22-07-2022	4	22
23-07-2022	1	34
23-07-2022	2	21
23-07-2022	3	0
23-07-2022	4	33

Is the fact "Quantity sold" additive, semi-additive or non-additive?

- ○ Non-additve
- ○ Semi-additive
- ◉ **Additive**

Good job!
Adding up any discounts across any dimension (either different products or different days) does not result in a meaningful number.

Question 3:

This table displays the discount of each products in a given day that the company offers.

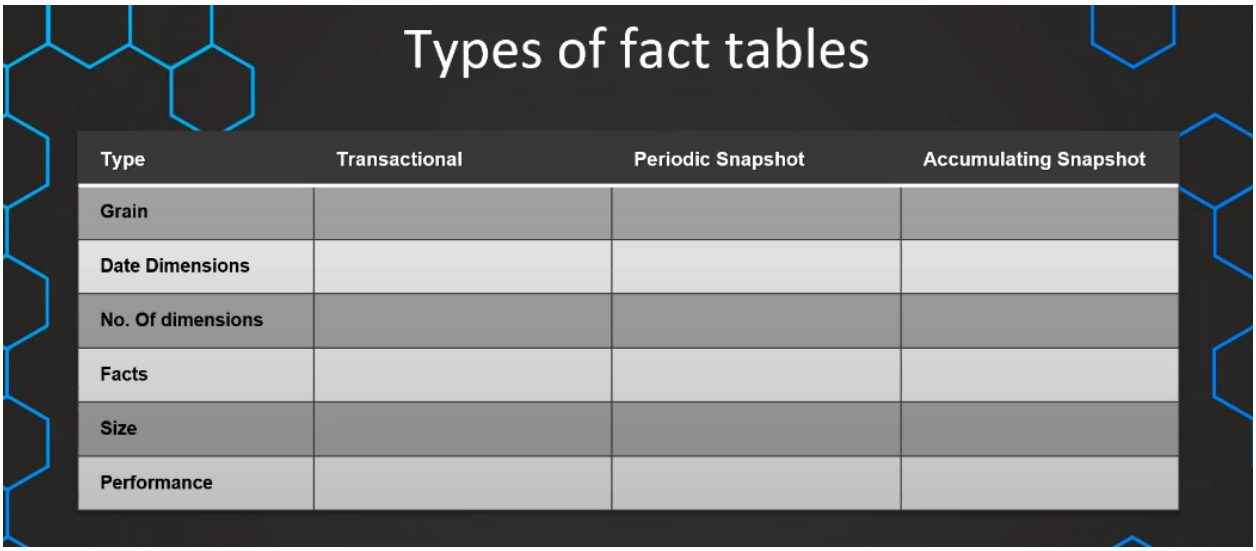
Date	Product_FK	Discount
20-07-2022	1	0%
20-07-2022	2	21%
20-07-2022	3	9%
20-07-2022	4	23%
21-07-2022	1	15%
21-07-2022	2	19%
21-07-2022	3	14%
21-07-2022	4	13%
22-07-2022	1	17%
22-07-2022	2	19%
22-07-2022	3	3%
22-07-2022	4	7%
23-07-2022	1	9%
23-07-2022	2	7%
23-07-2022	3	9%
23-07-2022	4	16%

Is the fact "Discount" additive, semi-additive or non-additive?

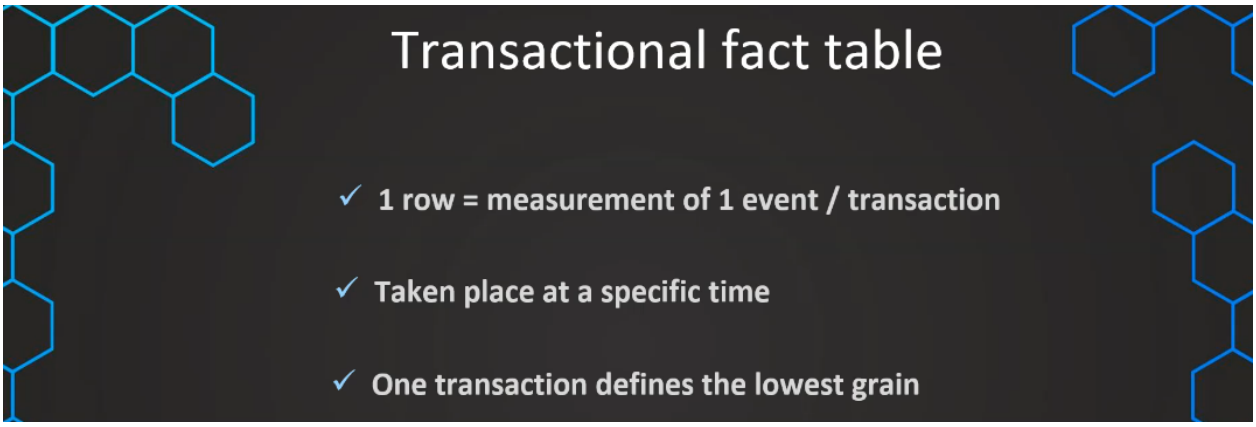
- (●) **Non-additve**
- ( ) Semi-additive
- ( ) Additive

Types of Fact tables

- we need to talk about the three different types of fact tables.
- And luckily, there are only these three different types.
  - Transactional
  - Periodic Snapshot
  - Accumulating Snapshot fact table.

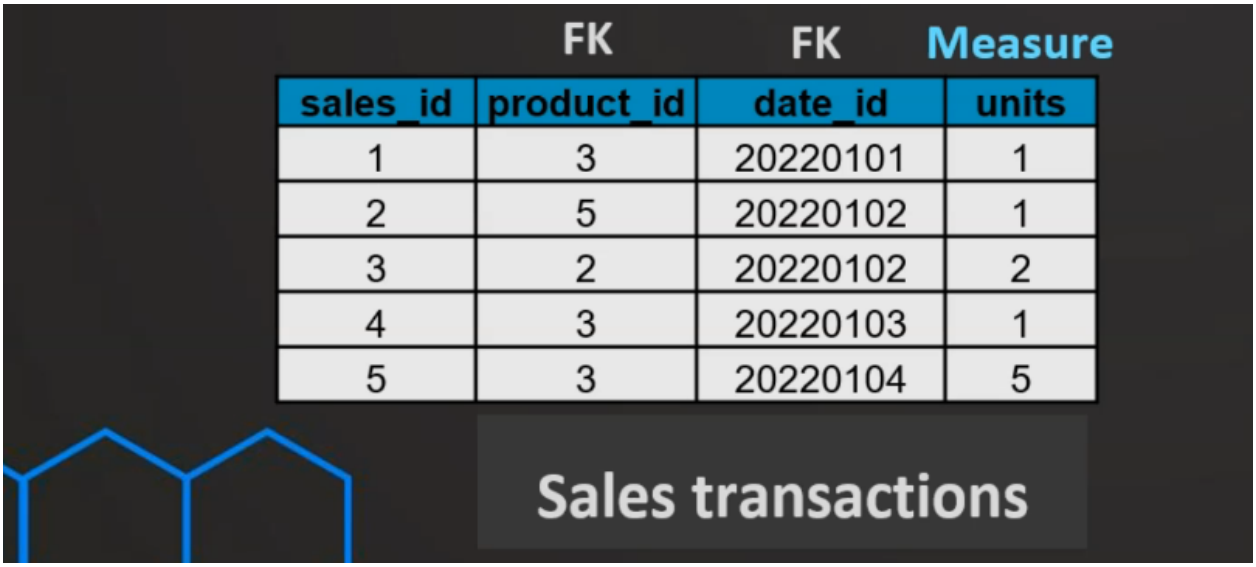


Transactional Fact table

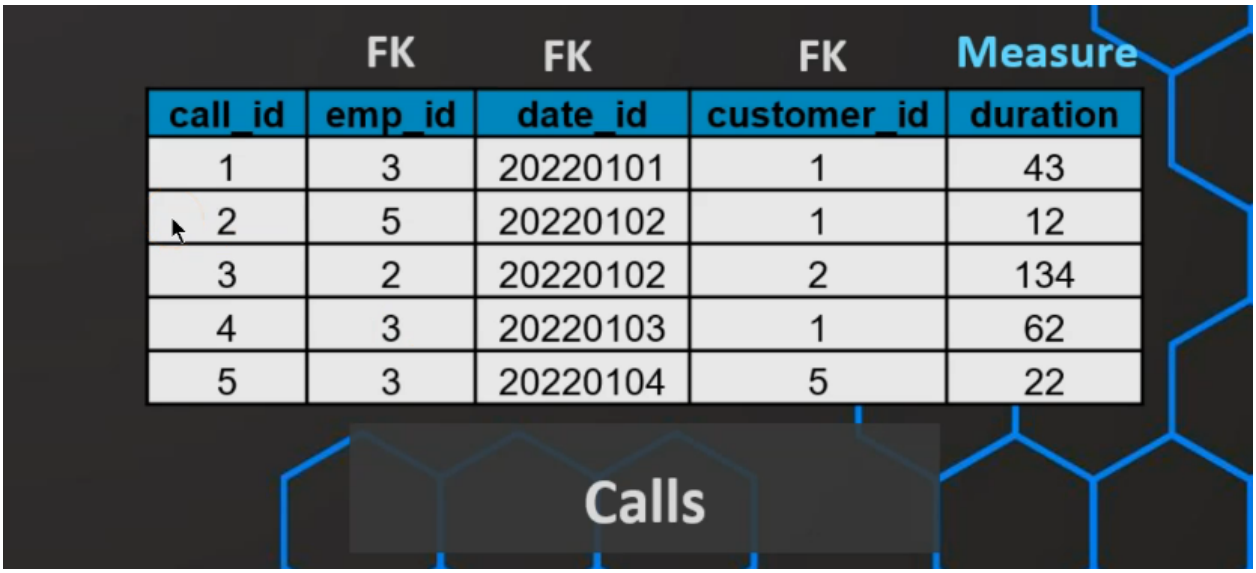


- The most fundamental fact table is the transactional fact table. So therefore, we first want to learn about this transactional fact table.
- In this transactional fact table, we have one row that is defined by one event or one transactions.
- So the facts in this transactional fact table are just measurements of one event or one transaction.
- So this is one transaction that takes place at one specific time and at a specific place usually. So this is something that is happening, an event or a transaction.
- For example, a sales transaction or some other examples

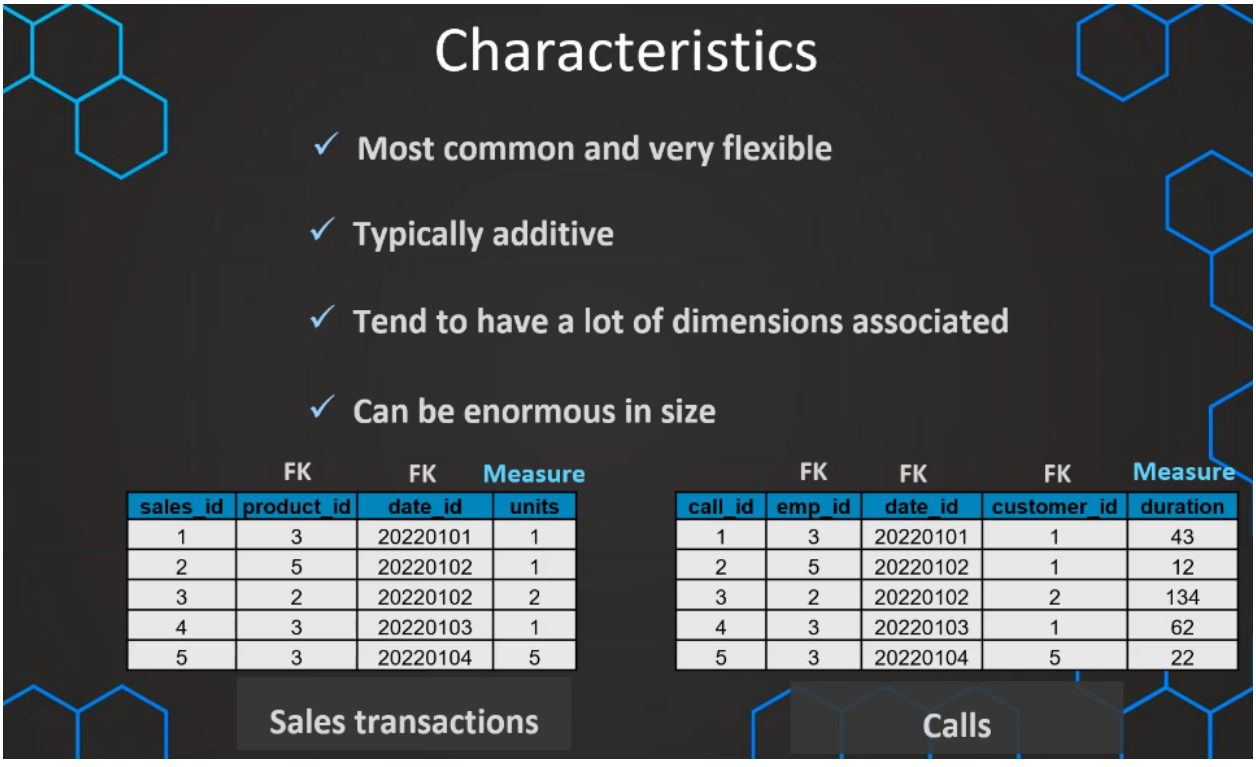
- And also this transaction is the definition, basically, of our grain.
- So our grain means we have one transaction that makes one row and therefore the transaction.
- So one transaction defines the grain. But now let's have a look at an example of that.
- So these are just very common examples. For example, we have a sales transaction.



- And in this sales transaction we see that one row is just one transaction.
- And in this transaction we can have different measurements. So for example, we have units. How many units have been sold in this single transaction? And then we have also foreign keys. So we have, what was the product? What was the time of the transaction? And also of course the date and other dimensions.
- And the same we can also apply for the other example of the calls where we have one call is basically like one event.

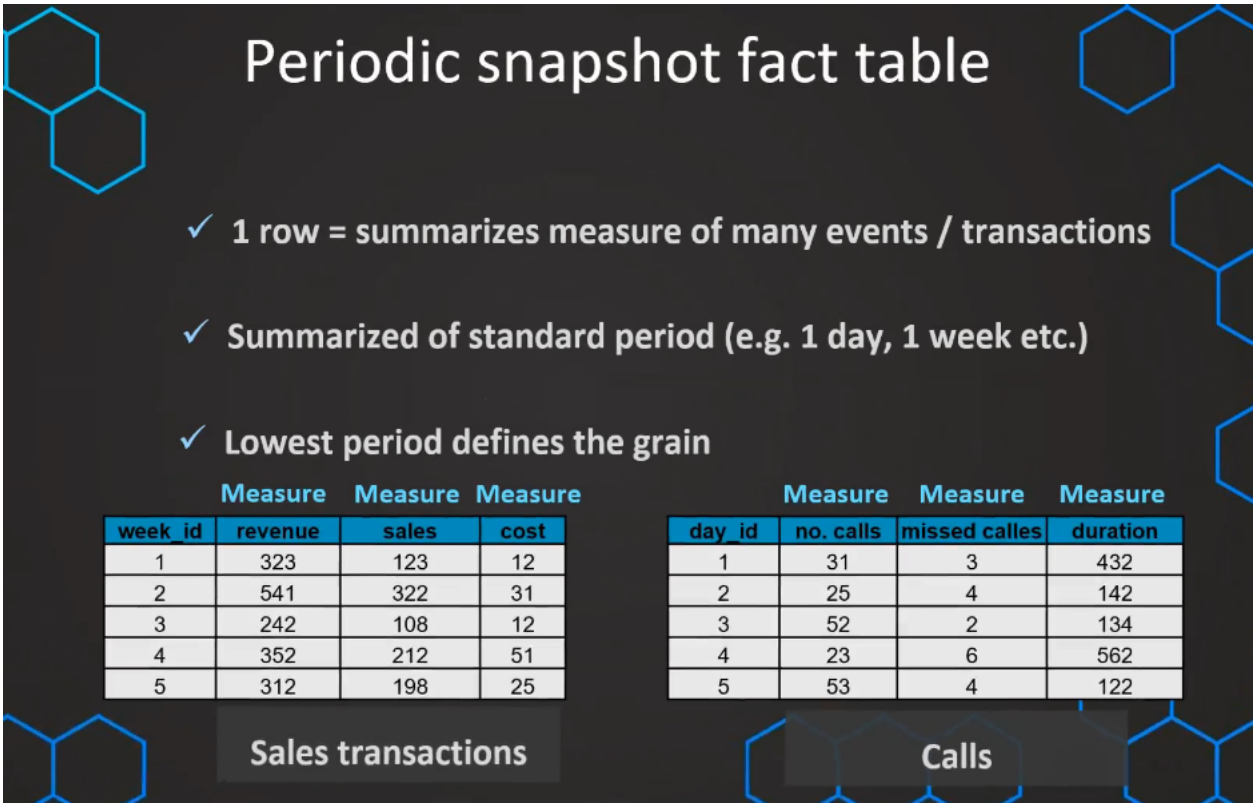


- And then we can have different measurements associated. So for example, how long was the call and other measurements of that event? And of course again, we have different foreign keys.
- But now what is specific about this transactional fact table? We can see that there are, for example, many foreign keys. And this is one of the important characteristics.

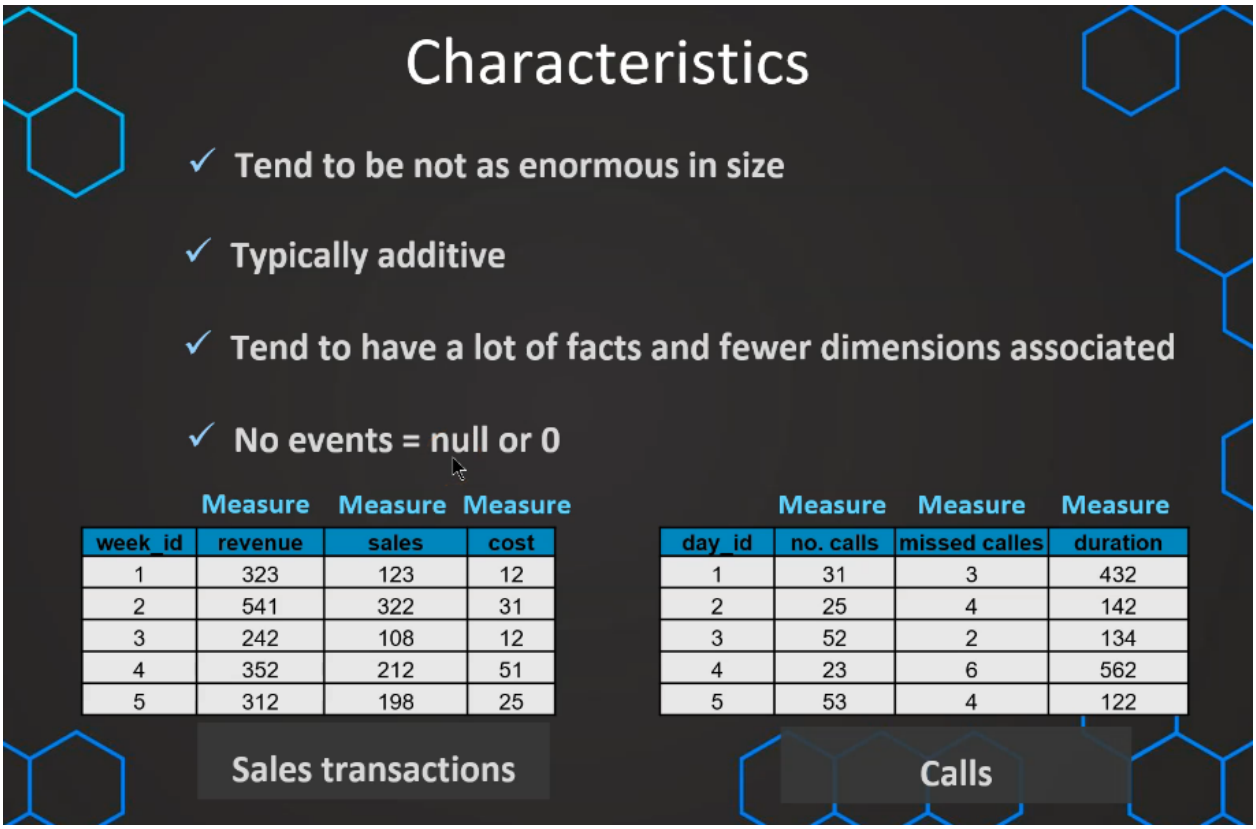


- But first we want to note that this is the most common type and it's also very flexible because we can analyze it in a lot of different ways with many dimensions.
- And also usually all of these transactions and these values that we measure, they are usually additive. So this gives us a lot of flexibility.
- And also what helps in different analysis is that those types of fact tables have usually many foreign keys associated. So they have a lot of dimensions that we can analyze.
- But one problem that we commonly have with those fact tables is that they can be enormous in size and their growth is also usually very rapid. So they can grow very fast.
- And therefore oftentimes we need to aggregate those tables. So this is the most fundamental fact table and if we've understood that, we can also easily understand the other types of fact tables.

Periodic Fact table



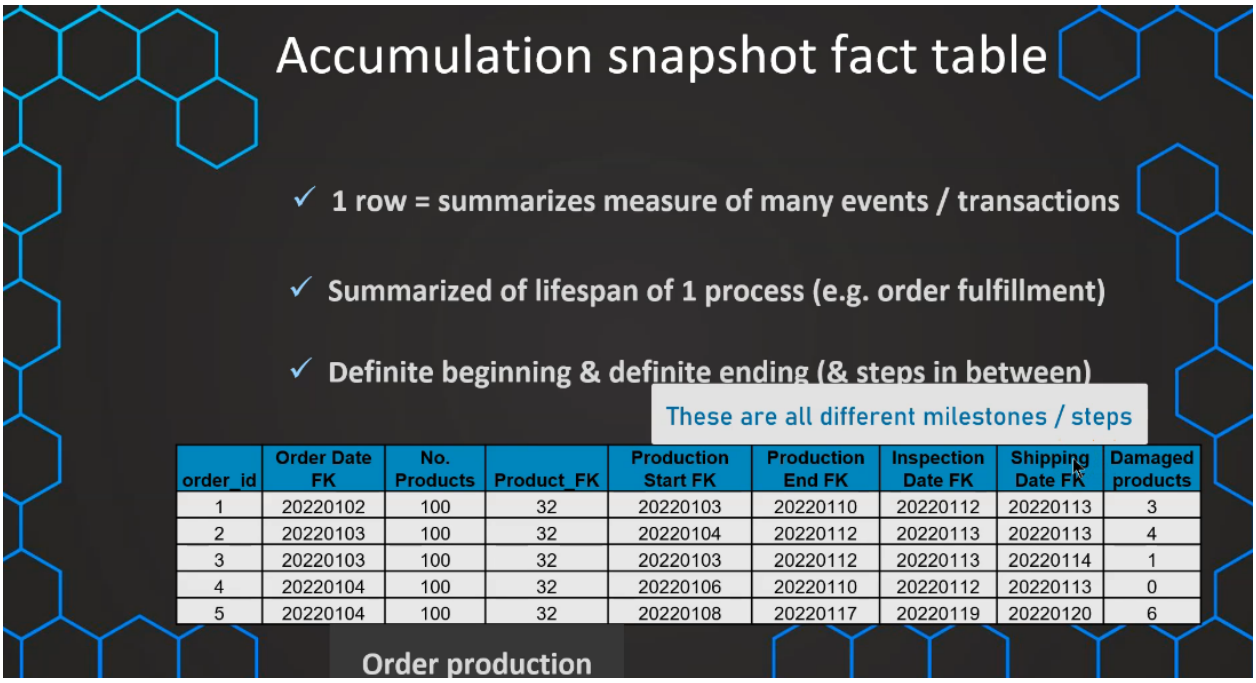
- In this fact table, we have one row defined by the summation or aggregation of a measurement across many events.
- So it is taking place in a standard period. So we aggregate all of the events, all of the transactions, and calculate the related measures for one specific standard period.
- So usually, this is one hour, one day, one week, one month, and so on, and then we have this all summarized.
- And of course, now this lowest period, so this standard period, is defining our grain, for example, one day, one week, or one hour. And an example of that is if we have, for example, our sales transactions.
- So we see that there's usually a transactional table behind or that is underlying this snapshot fact table, because we have now just one period, in this case, one week, and then at the end of the week, we just take a snapshot or we take the aggregations, how much revenue, how many sales, how much costs did we have? And there, you can see that they contain usually a lot of measures and not so many dimensions.
- And if we go also back to our calls example, we can see we have summarized the number of calls in a given day, the number of missed calls, and the total duration of our calls. And you can also see, again, many measures and usually not so many dimensions.



- Now, what is specific about this fact table type? This fact table type is usually not so large in size because we have standardized periods, and therefore, the grain is not so detailed, and we can see that there's already happening some aggregations, and therefore, this table is also not growing so rapidly, but it's growing very continuously.
- So we have always just one day or one week coming in addition to our data.
- So with that type of fact table, the growth of our fact table is very controlled, so it's not growing that rapidly. So this is one benefit, and of course, this can also help with the performance.
- And also still, this data is typically additive, and those are usually the grain levels that we need to analyze.
- So this is the defined grain that is just interesting for the analysis. So usually, we also don't lose so much analytical value, because we have defined our grain with a lot of thought.

- And also, we have seen that there are a lot of facts, usually, and not so many dimensions associated. And also we need to note if we have even a day or one period where there are no events, no transactions, then we can use either null or the zero.
- So if zero is really more representative, because we have zero sales, then the number should be zero.
- But for example, if we don't have any sales anyways on the weekend and we don't want to have the zero calculated for the average, then we can also just alternatively, for example, use a null value. So this is the second type of our fact table, Periodic Snapshot Fact Tables.

Accumulating snapshots



order_id	Order Date FK	No. Products	Product_FK	Production Start FK	Production End FK	Inspection Date FK	Shipping Date FK	Damaged products
1	20220102	100	32	20220103	20220110	20220112	20220113	3
2	20220103	100	32	20220104	20220112	20220113	20220113	4
3	20220103	100	32	20220103	20220112	20220113	20220114	1
4	20220104	100	32	20220106	20220110	20220112	20220113	0
5	20220104	100	32	20220108	20220117	20220119	20220120	6

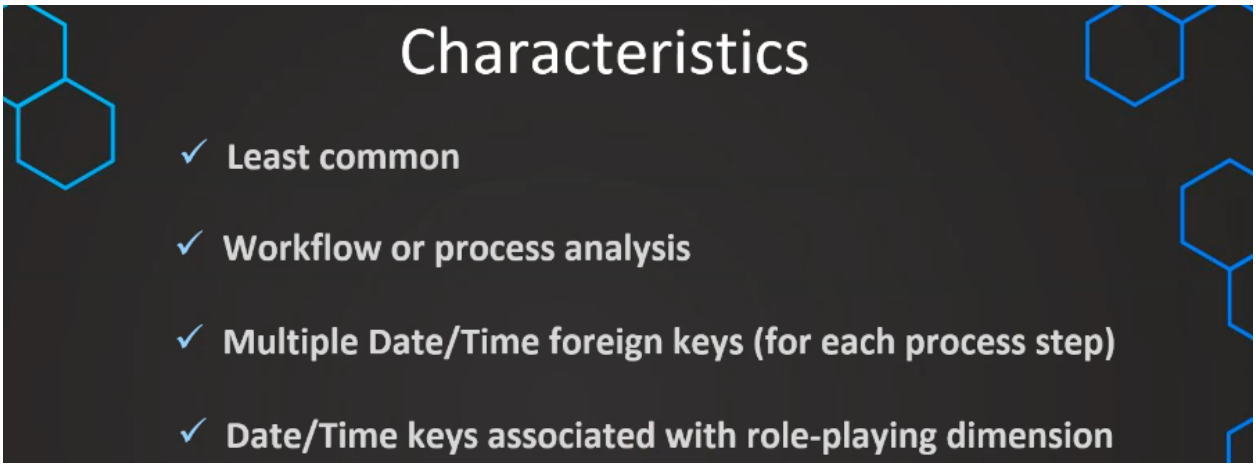
- Now let's talk about the last type which is the accumulation snapshot fact table.
- We've already learned about the periodic snapshot fact table and this one is a little bit similar except that we have one row in this table being defined by the summation of a measurement of many events again but now this period, it's not standardized, but it's defined by the lifespan of one process.

- So for example, one order fulfillment or any other process that has a specific beginning and a specific ending. And usually also some milestones or steps in between that are also interesting to analyze.
- So this helps if we want to have some workflow analysis or some process analysis.
- And an example of that could be, for example, as we've mentioned, order fulfillment or some order production process.
- For example, we have one order that is coming in at a manufacturer. We have the date foreign key. Then we have how many products have been ordered, what type of product. And then also we have many different date foreign keys for every single step or every single milestone that we have in between this process.
- For example, the start and the end of the production, the date of the inspection, the shipping date. And also for all of these in-between steps, we can have associated facts.
- So for example, how many damaged product have been found during the inspection? And also as we have seen how many products have been ordered.
- So you can see that it's typical for this type of fact table to have some measures but many date dimensions.

	Date FK	Measure	Date FK	Date FK	Date FK	Date FK	Date FK	Measure
order_id	Order Date FK	No. Products	Product_FK	Production Start FK	Production End FK	Inspection Date FK	Shipping Date FK	Damaged products
1	20220102	100	32	20220103	20220110	20220112	20220113	3
2	20220103	100	32	20220104	20220112	20220113	20220113	4
3	20220103	100	32	20220103	20220112	20220113	20220114	1
4	20220104	100	32	20220106	20220110	20220112	20220113	0
5	20220104	100	32	20220108	20220117	20220119	20220120	6

Order production

- So this date dimension is very common and we have usually a lot of them in the accumulation snapshot fact table.
- And what is characteristic about this type of table? It is the least common, but still it can happen. And therefore it is good to also know about it.



- So as we've mentioned, this is some type of table that is used and very good if the workflow or the analysis of the process is very important.
- So in that case, we can use such tables that are helpful in this process. And of course, we've also seen there are many date or time foreign keys for each of the steps in the process. And we'll also see later on that for this type of table with many date foreign keys, we still want to use only one single dimension.
- And then this will be called a role-playing dimension. We learn later on what that exactly means if we talk more about dimensions.
- But what we can take away now is that we use only one dimension and there are many possible connections then to this dimension table.

Comparing fact table types

Type	Transactional	Periodic Snapshot	Accumulating Snapshot
Grain	1 row = 1 transaction	1 row = 1 defined period (plus other dimensions)	1 row = lifetime of process /event
Date Dimensions	1 Transaction date	Snapshot date (end of period)	Multiple snapshot dates
No. Of dimensions	High	Lower	Very high
Facts	Measures of transactions	Cumulative measures of transactions in period	Measures of process in lifespan
Size	Largest (most detailed grain)	Middle (less detailed grain)	Lowest (highest aggregation)
Performance	Can be improved with aggregation	Better (less detailed)	Good performance

- Now that we've talked about the different types of fact tables, it's time to compare them with each other.
- In the transactional fact table, one row is defined by one transaction. In the periodic snapshot, we have one row that is defined by the aggregation of one period. And also we can have multiple dimensions that add also to the grain.
- So for example, we can have one period, so one day, and all of the regions, so the sales of all of the regions, and those additional dimensions can also add to the grain.

day_id	region_id	sales	cost
1	1	123	12
1	2	322	31
1	3	108	12
2	1	212	51
2	2	198	25
2	3	213	43

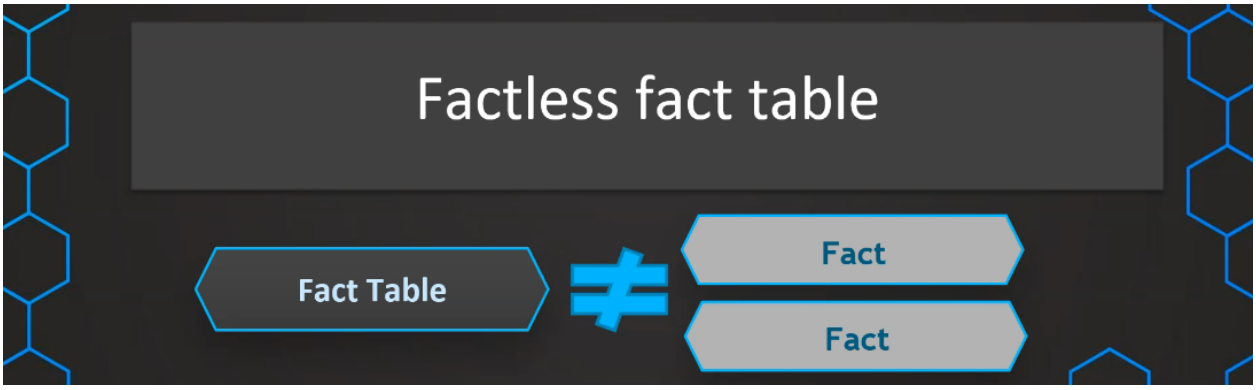
- And in the accumulating snapshot, we have one row that is defined by the lifetime of one process or one event. And those tables also differ in the date dimension.
- In the transactional table, of course, the date is the transaction date. And in the periodic snapshot, we have that the snapshot date is our related date dimension or foreign key. So this is always the end of the period.
- And for the accumulating snapshot, there are of course then, just multiple snapshot dates, that are always related to the end of the period
- . And we've learned also that in the transactional fact table, this is very dimensional, which helps us to analyze our data very well.
- And in the periodic snapshot, we've seen that in this case, usually the amount of dimensions is a bit lower, but we have the data just a bit more aggregated and therefore the size is also a bit smaller.
- And in the accumulating snapshot, we've seen that we tend to have, especially, a lot of date foreign keys, and also the dimensions are therefore

very high.

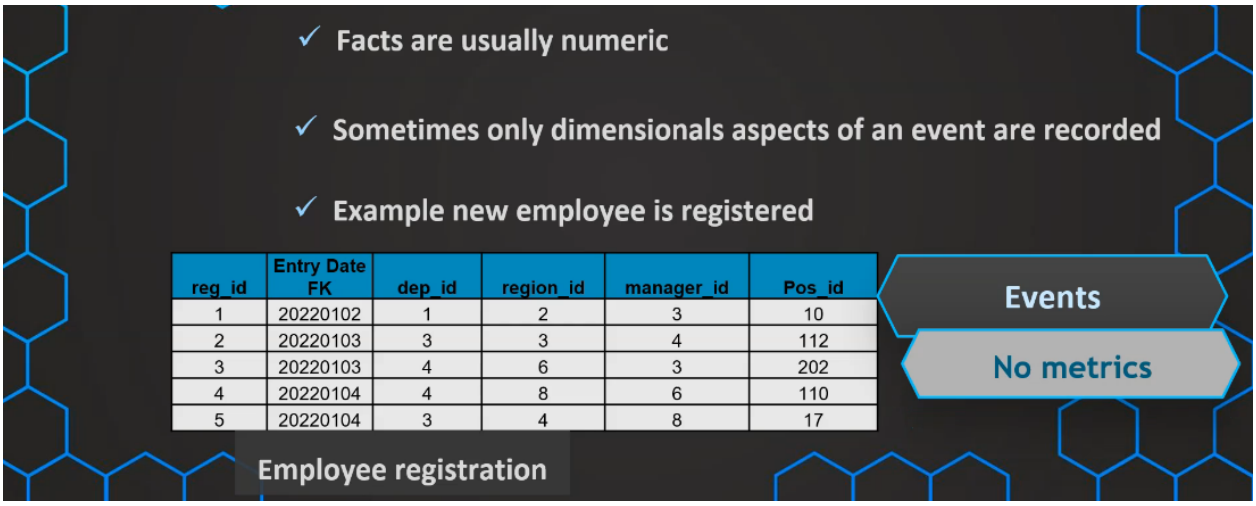
- And now the question what are the measurements of facts in the different tables? Of course, in the transactional table we want to measure the performance of the transactions. And then the same goes for the periodic, except that we have cumulative measuring, because we just aggregate all of that measures in a specific period. And the same goes for the accumulating snapshot table in here, we also just measure all of those processes during the lifetime.
- Note, also, that the transactional fact table is usually the largest because it has the most detailed grain, and therefore it is very flexible, but the performance can be sometimes improved, usually it is a good performance as well but it can be improved.
- And oftentimes there's the need for some kind of aggregation for our analytical purposes.
- So in the periodic snapshot table, basically, we have done that already because we have defined our grain, and then we have aggregated the data.
- So therefore, the performance is a little bit better in the periodic snapshot. But usually nowadays, we have very good databases with a high performance, and therefore, oftentimes, we have no problem with the performance.
- But if we have some problems with the performance, it can help to just aggregate the data, define a grain. And this is also something that we learn a little bit more about later on. But now we have learned about these three different fundamental types of fact tables.

Factless Fact table

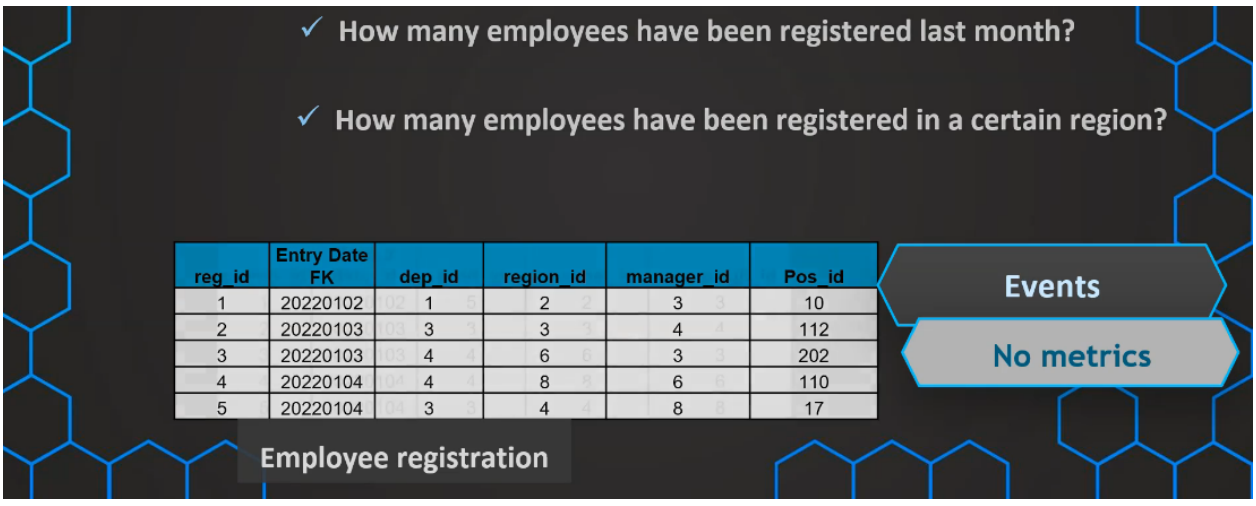
- There's also one type that is a little bit special, and this is the so-called factless fact table. And this is not a contradiction, because we know that a fact table is different from a fact.
- So in one fact table, there can be multiple facts. So the fact itself is just the numerical measurement that is used to track the performance of a certain business process. And the table is the entire table itself that keeps track of all of those facts, foreign keys, and everything included in this fact table.



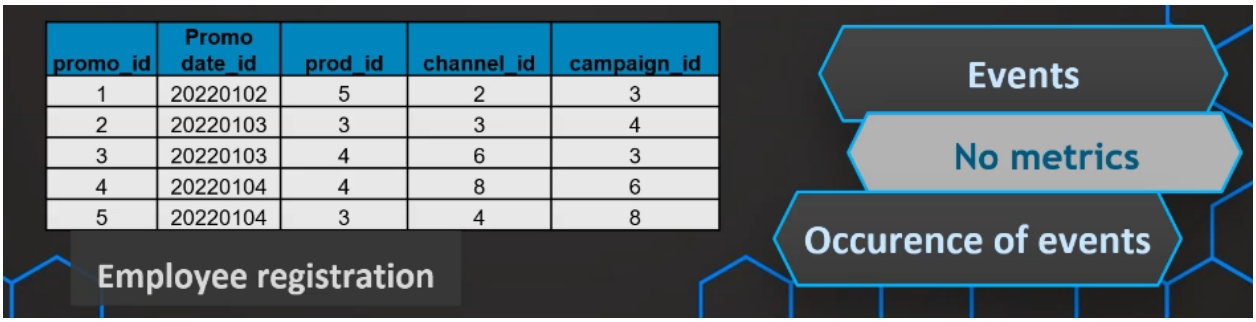
- And those two things are now not the same. And it is actually even possible that a fact table has no single fact at all.
- So we've learned that fact is usually a numerical value, so a measurement that tracks some kind of performance.
- But sometimes in some fact tables, only the dimensional aspects of a certain event or a certain transaction is recorded.
- So in this case, we don't have any facts, and we have basically a factless fact table. So this is something that is possible as well.
- But now let's have a look at the following example.
- We are working in a company and there is recorded every new employee that is registered in the company. So therefore we have a table, and this is our fact table.
- So we record every single event, so every single registration and all of the dimensional aspects. So what is the entry date of this employer? What is the department that this new employee is registered in, and what is the region, what is the manager of this employee, and all of the associated dimensional aspects.



- And there are, in this case, no metrics, but all of the dimensional aspects just recorded.
- And with that, we can now answer questions like, how many employees have been registered last month? We can do that very easily in SQL just by filtering on the month with the dimension. And then we can also just count the number of rows.
- Of course, also, we can do that in all of the BI tools. We just filter by the date, for example, and then just count all of the number of rows.
- So like this, we can also answer questions like how many employees have been registered in a certain region, in a certain department, and so on.



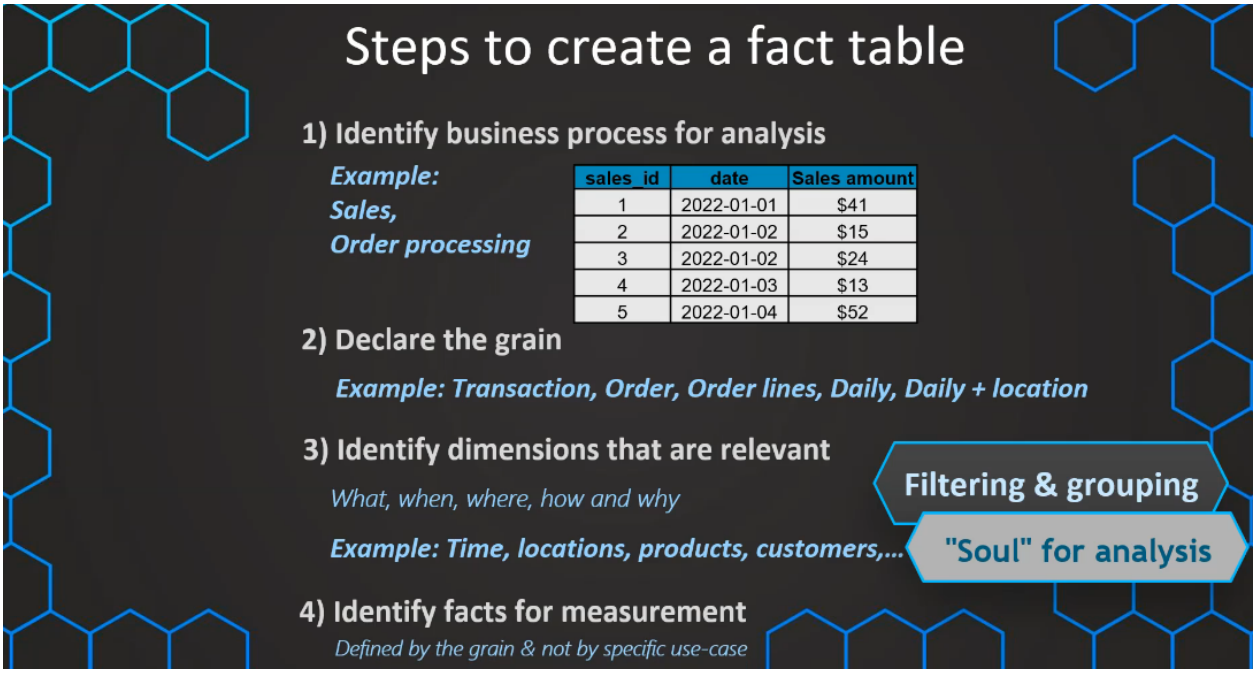
- So these are all of the possible use cases that can be pictured with a factless fact table.
- So whenever we just want to record the events without having any metrics, then we can still have a factless fact table, which is not a contradiction.
- Another example is if we have certain promotions and we have no metrics associated with them, so it's just an event with a certain promo code, a certain product that is promoted, a certain campaign that is associated with this promotion.



- And then we can keep track of those events and structure that in a fact table, because those are events that are happening.
- But we just don't have any metrics associated because we just register and record the occurrence of the events and the associated dimensional aspects.
- And now that we've learned about the different types of fact tables, we want to have a closer look at the steps that we need to take when we want to design and implement our fact tables.

Steps in designing fact tables

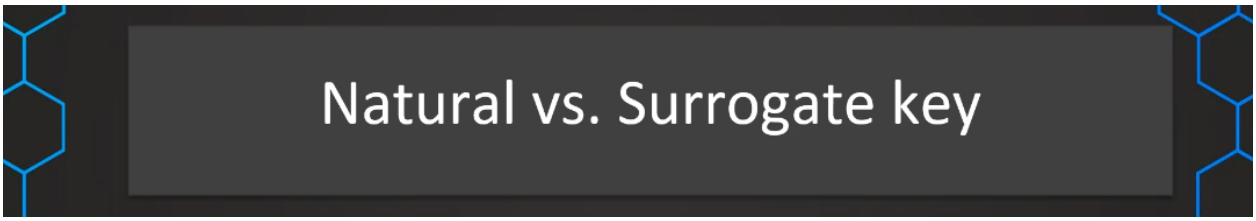
- Usually there are a few key decisions that we need to take when we want to design our fact tables.
- And basically, or luckily, there are just four key decisions that we need to take when we want to design our fact tables.
- And we take those key decisions by answering questions about our business needs. So when we want to make those decisions, we need to consider the business needs, and with that we can answer or take those key decisions, and then from that we can design our tables and our columns.
- So we have basically laid out the design of our fact table.
- Now what are these key steps that we need to take?



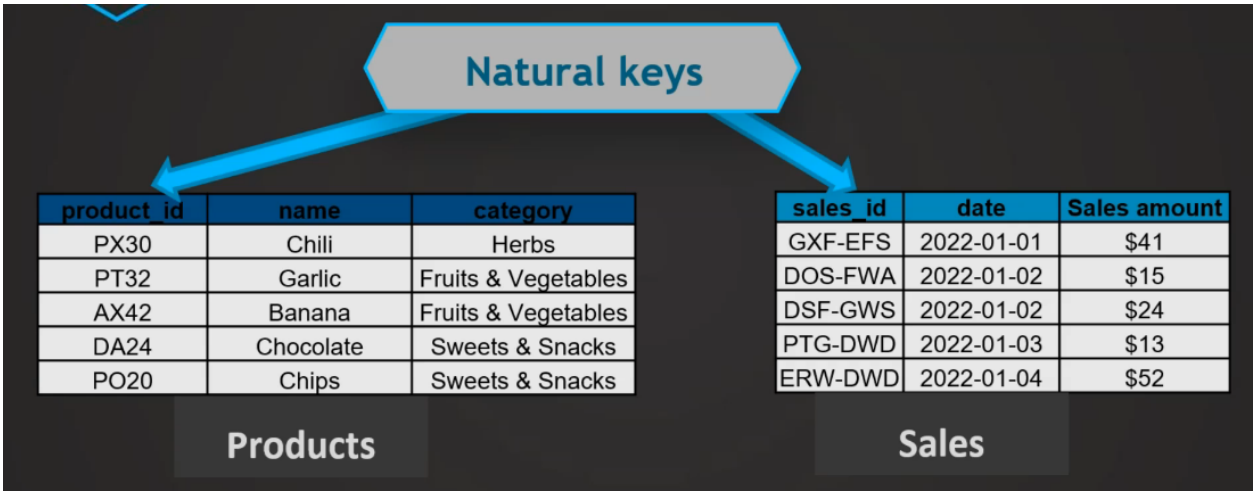
- First we want to start with identifying the actual business process that we want to analyze.
- For example, this can be we want to analyze our sales, we want to analyze the order processing, the order fulfillment, or anything else that is a business process that we want to analyze. For example, as we've seen in here, this can be a sales table, and we have the different aspects included in here.
- Now the second step that we need to take when we want to design our fact table is to define the grain. We've already talked about the grain. This is basically the level of detail in our table.
- This is a crucial decision because this is a important aspect in our analysis. And the grain, as we have learned, is just defining what is one row referring to.
- So one row is equal, basically, to one transaction, one order that has taken place, or also we can have, as we've seen with the periodic snapshot fact tables, some periods as a grain,
- For example daily, or also a combination of daily and a location, so every day for every certain location. And It is also recommended to rather go with a finer grain, so with a higher level of detail.

- So usually if we have every single transaction, and it is not pre-aggregated, this is a better option, a better path to go, because this will leave us open for all the different aspects of analysis.
- And we don't need to make any presumptions with pre-aggregated data, because if we have pre-aggregated data, this will limit the way we can analyze this data later on.
- And we have most of the options available if we go with a more atomic transactional grain, and then later on we can still aggregate the data for our data marts for specific use cases if we have those specific use cases in our data marts.
- So therefore, if you want to define the grain, you should rather use a more fine grain, so a more atomic level with a higher level of detail.
- The next step would be to identify the dimensions that are relevant. And we can do that by just going to the questions, what is happening, when it is happening, where, how, and why.
- So if we focus on these words then we can also find our different dimensions. For example, it can be different time aspects, locations, products, customers, and so on, whatever is important in our business scenario.
- So these are the dimensions, of course, that are giving us the option to filter and group our data. So basically they are the entry points for our data analysis, and therefore we also refer to them sometimes as the soul of our data warehouse for our data analysis.
- And now the last step is just to identify the facts for our measurements, and then we can aggregate them if necessary for the defined grain. So these are the steps that we want to take, and these are the steps that we should take if we want to design and lay out our fact tables.
- And later on we'll also focus a bit more on the ETL aspects, but it's always good to keep those steps in mind when we are designing our fact tables.

Surrogate keys

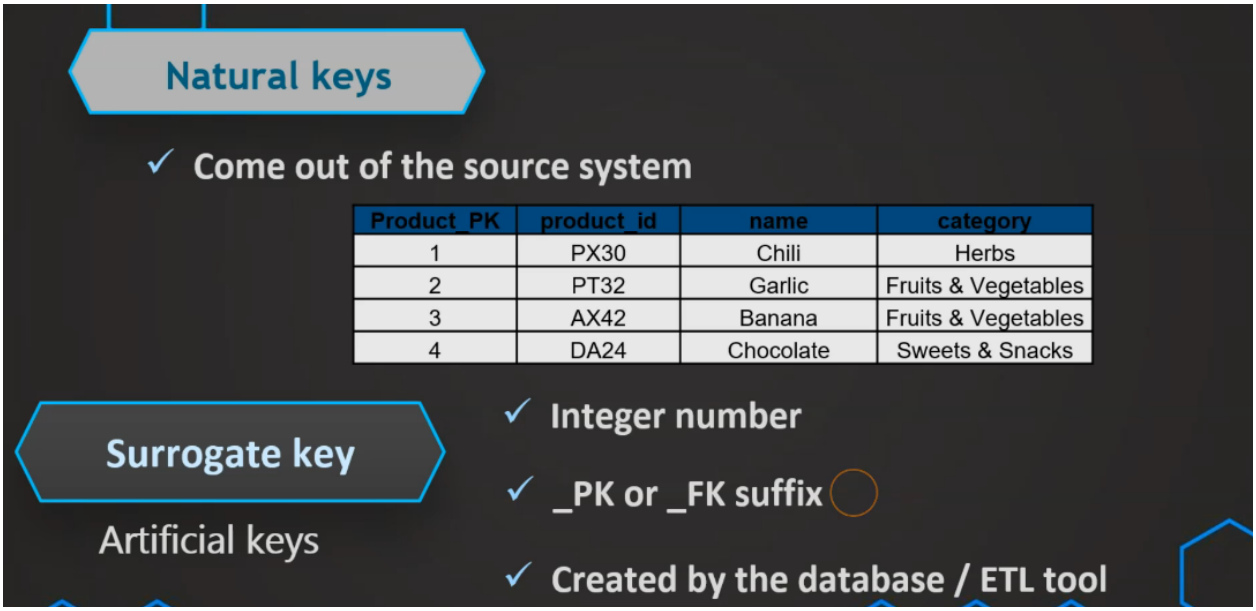


- Talk about the difference between natural and so-called surrogate keys.
- So this is what we want to understand, and also then get some guidelines on how we should use them.
- So first of all, let's understand what are natural and surrogate keys. So if we have a look at our product dimension, this is the data, how it might come out of our system.

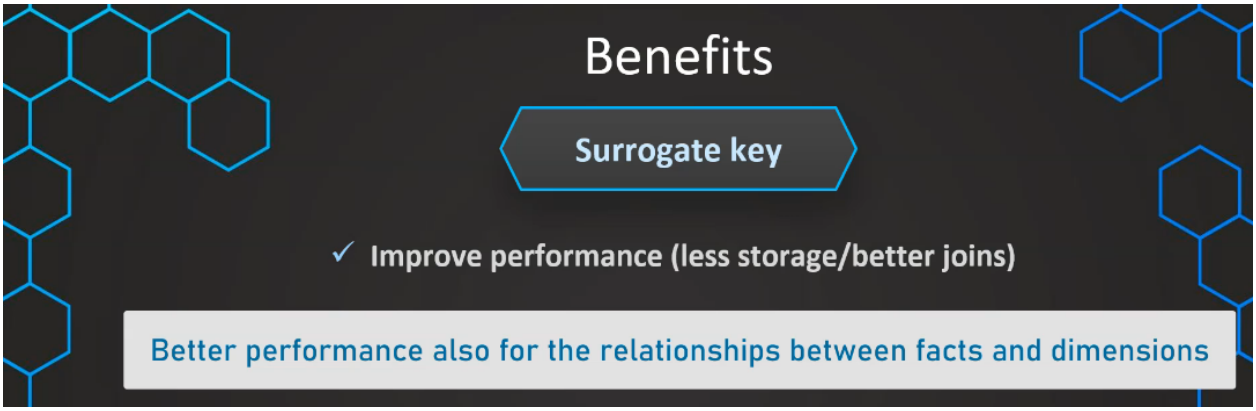


- And if we have a look at the product ID, we see that this is a alpha numerical number, and they can be usually quite bulky and not really nice to handle.
- And they have a couple of disadvantages associated with them.
- Also, if we have a look at a transactional table, we can again see a bulky, and it can be even a lot larger number. And this is just referring to the transaction ID basically that is generated by the source system.
- And now this is not the ideal way of how we have things managed in our data warehouse.
- And those are now our natural keys, but we don't want to usually leave it that way.

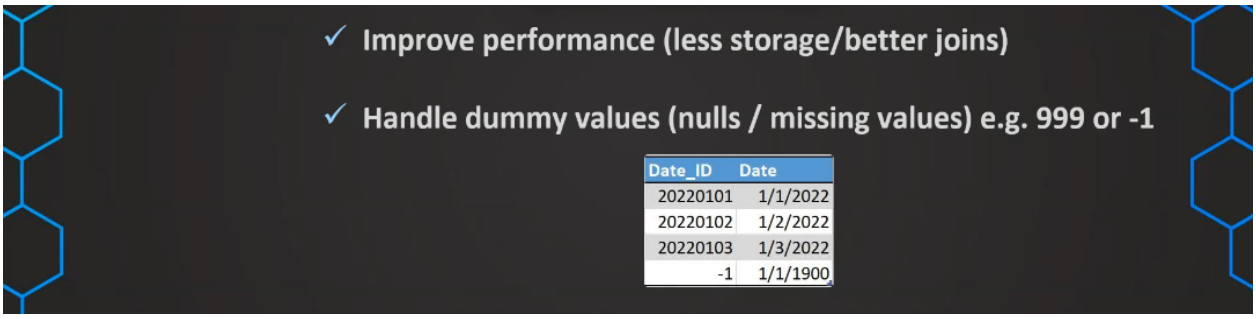
- So we have learned a natural key, it's just coming out of the source system.
- But now we can also generate our surrogate keys. And this is then nothing but a artificial key.



- And therefore, it is sometimes also just called an artificial key. And like this, we can see that it's opposed to the natural key because we create those surrogate keys in our ETL process.
- They are just very simple integer numbers. And also to understand that those are actually our surrogate keys, we have usually the suffix PK for primary key and FK for the foreign keys.
- And with those suffix, we can then usually immediately recognize that this is the surrogate key.
- And as mentioned, they are just created in the database or by the ETL tool. And also this process is very simple usually. So it's not a lot of additional work, but we have quite some benefits with that.
- So let's talk about what are actually those benefits, and why should we create always those surrogate keys.

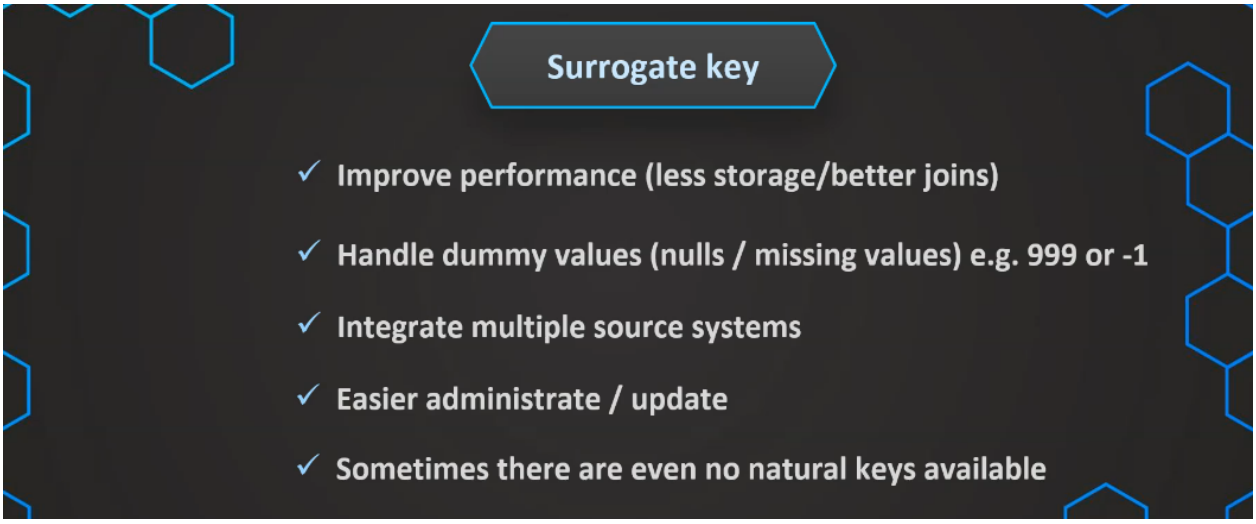


- So they are, as we've seen, much smaller in size. So we have with the natural keys, those very long and bulky alpha numerical strings. And if we just use the very small integer numbers, which can be just four bits in size, we have much less storage.
- And also the performance with that, if we use that column as an index, it's much better in the performance.
- And also, of course for the joins, those numbers work much better with a higher performance than those alpha numerical keys. And also, we can handle dummy values a lot better.
- So if we have some missing values, for example, a date is not existing, then we can just simply create negative one or another very high number maybe that is just referring to our dummy values, which is just signaling that there is no value available. And we can be more consistent with that.

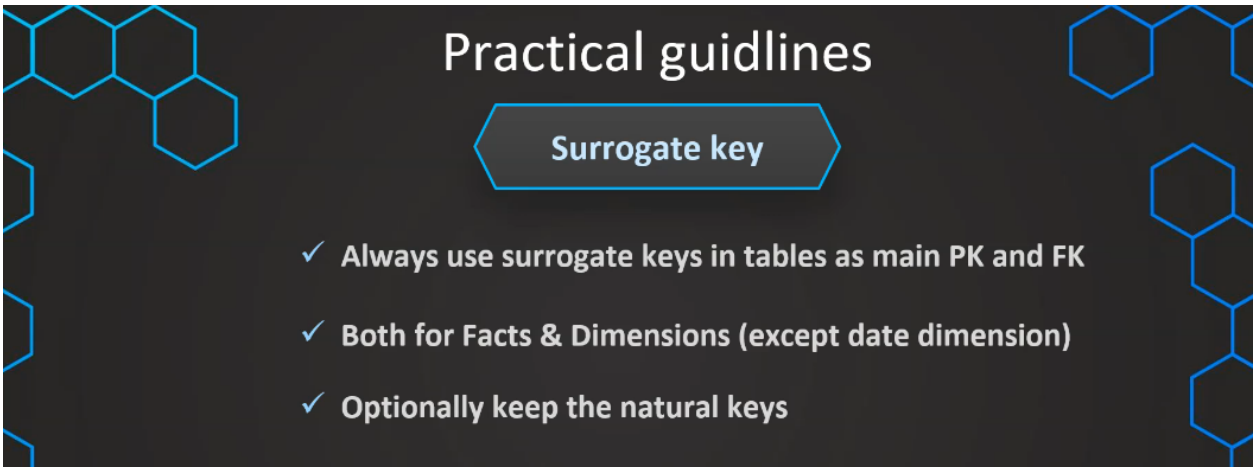


- But also it's very practical if we have multiple source systems that all have different numbers and sometimes they can be then duplicates.

- If we have two systems and they both use the same number, then it's very good to use those surrogate keys to integrate those multiple source systems.
- And in general, it's much easier to administrate and update those values. And we also see that if we talk about the slowly changing dimensions that it's better to have those surrogate keys for updating values, for example, in our dimensions.
- But also we sometimes don't even have a natural key available. And then it's also necessary to just auto generate those surrogate keys in our database or with our ETL tool.



- So this is just something that we should usually implement.
- We should always use those surrogate keys in our tables if we have our primary key and of course also for our foreign keys. And in general, It is recommended that we use those surrogate keys, both for fact and dimension tables.



- One exception is the date dimension. In the date dimension, we don't necessarily need to do that because it is very predictable, and we can just also stick with our date key. So this works in this case as well.
- But for all of the other dimensions, we should always use surrogate keys. And now the question is also do we keep our natural keys? And yes, in some cases if we want, we can decide to keep them.
- If we have it, for example, in the dimensions, then it is usually not a lot of data. And then it's usually not a problem to just keep the natural keys as well, just in case we might need them.
- Usually we don't need them, but it is if it's not a problem, we can also keep them. So this is something that we can do as well. So these are the very simple practical guidelines.
- So to summarize, a surrogate key is created in the ETL process by assigning just an integer number to every single row. And this is very helpful for us, and we should always use those surrogate keys in our tables.

Quiz

Good job!
The data is aggregated periodically (each day = standard period).

Question 1:

This table displays the number of items that are in stock in the company's warehouse for each product in each day.

Date	Product_FK	Items in stock
20-07-2022	1	15
20-07-2022	2	1
20-07-2022	3	18
20-07-2022	4	6
21-07-2022	1	34
21-07-2022	2	14
21-07-2022	3	34
21-07-2022	4	10
22-07-2022	1	27
22-07-2022	2	14
22-07-2022	3	31
22-07-2022	4	12
23-07-2022	1	13
23-07-2022	2	4
23-07-2022	3	15
23-07-2022	4	8

What type of fact table is it?

- ○ Transactional fact table
- ◉ **Periodic Snapshot fact table**
- ○ Accumulating snapshot table

Good job!
Yes, one row represents one sale (=transaction).

This was discussed in Lecture 32: **Types of fact tables** >

Question 2:

Below you see the star schema of a data model with dimensions and facts.

Dim_Date
Date_ID
Year
Quarter
Month
Day

Fact_Sales
SalesID
Date_ID
Quantity
SalesAmount
Item_ID
Location_ID
Brand_name
Category_name
Location_id

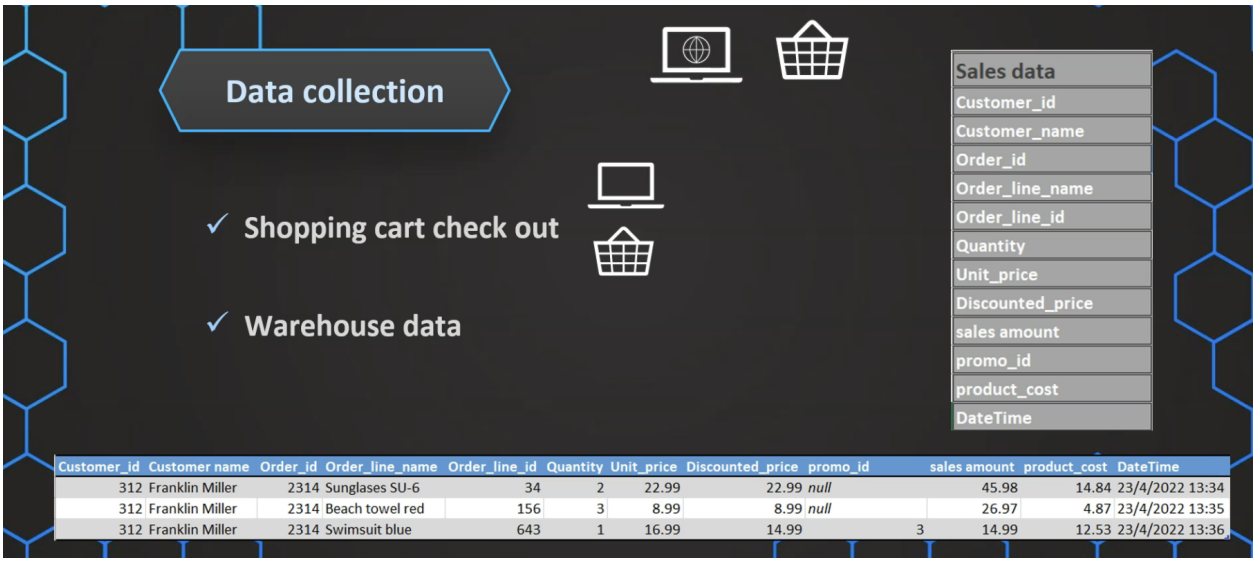
Dim_Location
Location_ID
Country
State
City
Location_manager

Dim_Item
Item_ID
Item_Name
Brand_name
Category_name

As what type of fact table would you identify the Fact_Sales table?

- (●) **Transactional**

Case study: The project



Customer_id	Customer name	Order_id	Order_line_name	Order_line_id	Quantity	Unit_price	Discounted_price	promo_id	sales amount	product_cost	DateTime
312	Franklin Miller	2314	Sunglases SU-6	34	2	22.99	22.99	*null*	45.98	14.84	23/4/2022 13:34
312	Franklin Miller	2314	Beach towel red	156	3	8.99	8.99	*null*	26.97	4.87	23/4/2022 13:35
312	Franklin Miller	2314	Swimsuit blue	643	1	16.99	14.99	3	14.99	12.53	23/4/2022 13:36

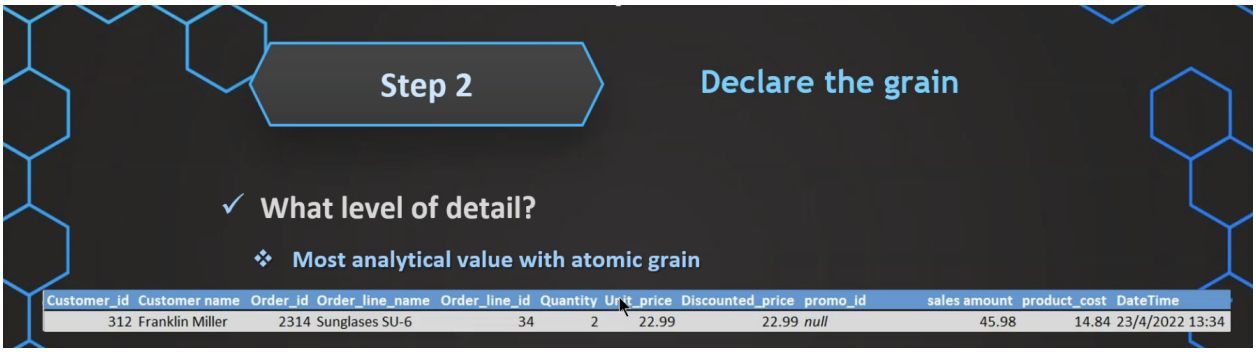
Case study: Step 1 - Identify the business process

- The first step when we want to design our fact tables, which is the basis, basically, of our data warehouse, is we need to identify the most relevant business process in our case.
- And in our case, it is the first data warehouse in the company, and therefore the question is, what could be a good and suitable business process for our first data warehouse? And in general, the guideline is that we should pick, of

course, the most crucial business process in the company for the business, this is in our case, the sales transactions.

- And also we should keep in mind data availability and data quality.
- So if the data quality is not so nice and it's difficult to access the data, then we should also, for our first data warehouse, keep other use cases also in mind, of course, depending on how critical this use case or this business process is for the company.
- Therefore, this is our first process in which we can analyze things like what products have been sold, what is the profit we have made in our sales in our different categories on our different websites, and we want to analyze the overall sales performance, the profit performance, on the different date-related features on the different dimensions that we are going to create, and, of course, also the performance over time.
- And this is now the business process and the different aspects that should be in the end possible to be analyzed. And with that business process identified, we can now move on to the next step in our process, and that is to declare the grain in designing our fact tables.

Case study: Define the grain

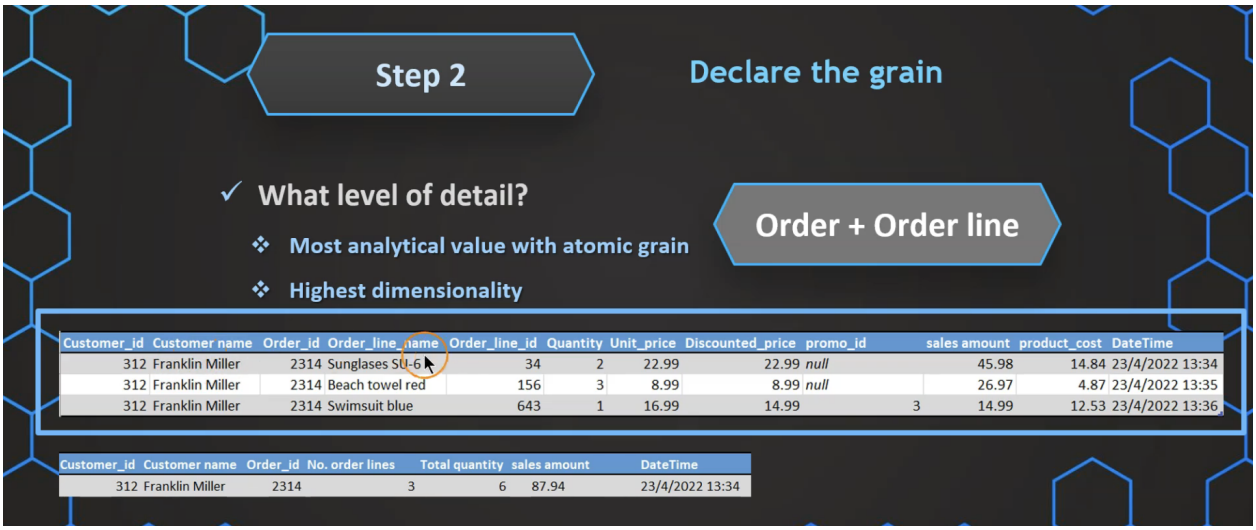


- Now that we've already identified the most important business process for our data warehouse that we want to use, we are now ready to go to step two, which is to declare the grain of our fact table.
- And the grain is referring to the level of detail. So we want to find out what one row of our fact table should represent.

- And here it is important to underline that the highest analytical value comes if we use the atomic grain. That means we use the highest level of detail in our fact table.
- So in our case, this would be that one row is representing one single item order line in a given order.
- So basically, one item that is in our cart of the customer. So this is the highest level of detail, and this is then a transactional fact table.

Customer_id	Customer name	Order_id	Order_line_name	Order_line_id	Quantity	Unit_price	Discounted_price	promo_id	sales amount	product_cost	DateTime
312	Franklin Miller	2314	Sunglases SU-6	34	2	22.99	22.99	*null*	45.98	14.84	23/4/2022 13:34

- And we do that because this brings us the highest dimensionality. So in our case, as I mentioned, this is the order and the order lines. So one order line in a given order.



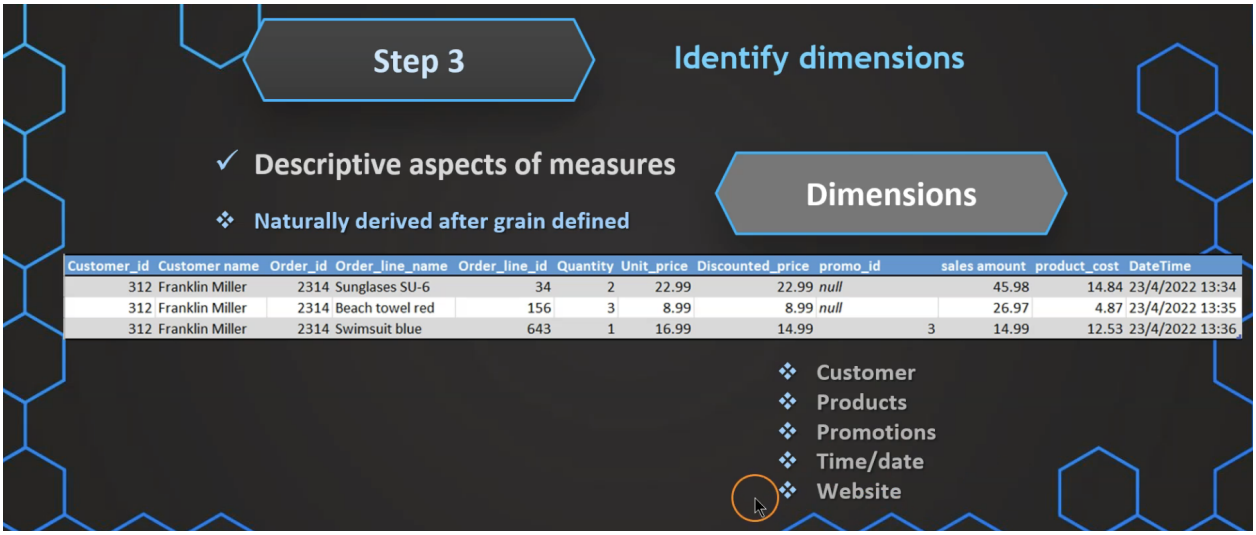
Customer_id	Customer name	Order_id	Order_line_name	Order_line_id	Quantity	Unit_price	Discounted_price	promo_id	sales amount	product_cost	DateTime
312	Franklin Miller	2314	Sunglases SU-6	34	2	22.99	22.99	*null*	45.98	14.84	23/4/2022 13:34
312	Franklin Miller	2314	Beach towel red	156	3	8.99	8.99	*null*	26.97	4.87	23/4/2022 13:35
312	Franklin Miller	2314	Swimsuit blue	643	1	16.99	14.99	3	14.99	12.53	23/4/2022 13:36

Customer_id	Customer name	Order_id	No. order lines	Total quantity	sales amount	DateTime
312	Franklin Miller	2314	3	6	87.94	23/4/2022 13:34

- So this would look like this. We have already seen that one row is representing one single item order line in a given order.
- For example, if we would not use that grain but we would aggregate the data a little bit and use the grain of a order, so then one row is just representing one order.
- We can say that we have now three order lines in that given order, the total quantity of products is aggregated to six.

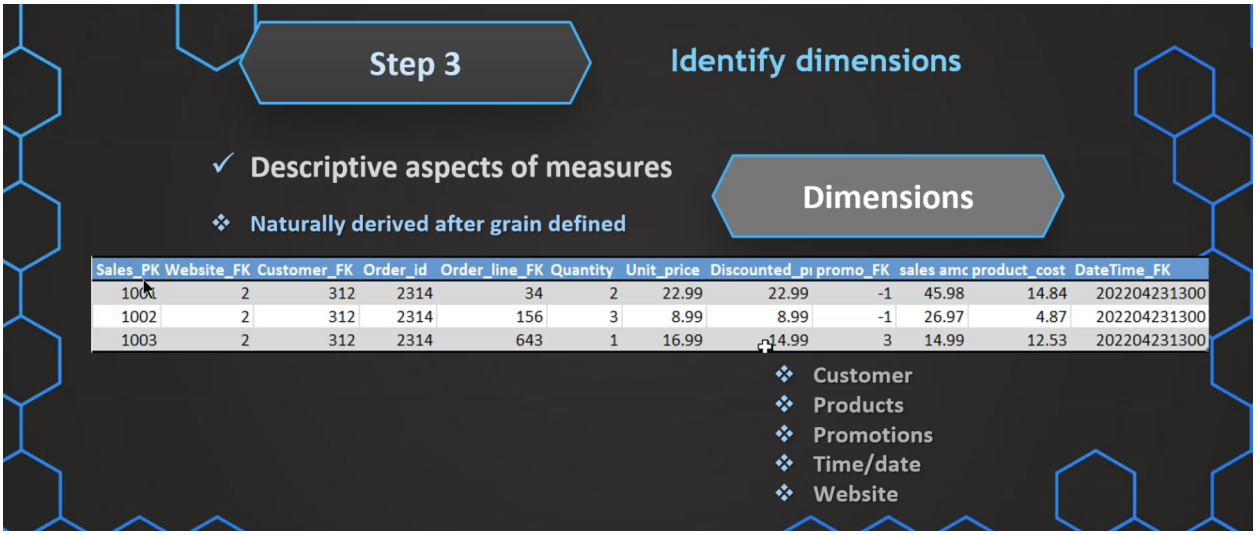
- The total sales amount is aggregated to around $88.
- And this is now not really ideal because now we see that we lose a lot of dimensions that are associated to another level of detail, so to a more atomic grain. A
- nd therefore it is recommended to really go with the atomic grain, in this case, the transactional level of our data.
- with that, we have then the highest analytical value. And also the performance is usually not a big problem. The databases usually nowadays have a good enough performance to process that amount of data very easily.
- And with that, we are open for all of the possible use cases. And with that, we have the highest analytical value in our data.
- And now that we have declared the grain, we want to continue with the next step, which is to define the dimensions.

Case study: Identify the dimensions



- Now that we have already identified the business process and declared the grain, we are now ready to find the dimensions.
- So basically, this is the question of describing our measures.

- So how are the users describing the measures? What is the descriptive context? So what are the stores, what are the locations, what are the products, categories, websites, and so on.
- And they are very easily and naturally now derived from the grain that we have defined.
- We have seen, we use the grain of the order line and the order and now all of the dimensions that are associated with that grain can be now derived.
- So for example, in our case, we can see that the derived dimensions are now the customers.
- We have also the products, so the order lines. And also we have then additional things, like the promotion and also date and time-related features.
- But also, we should not forget that even though in this case, the website is not included, we see that from which system is the data coming from.
- So actually, we also have this last definition of the website. So this comes, remember, from three different websites, so from three different systems. And with that, we get also the website where this data is coming from.



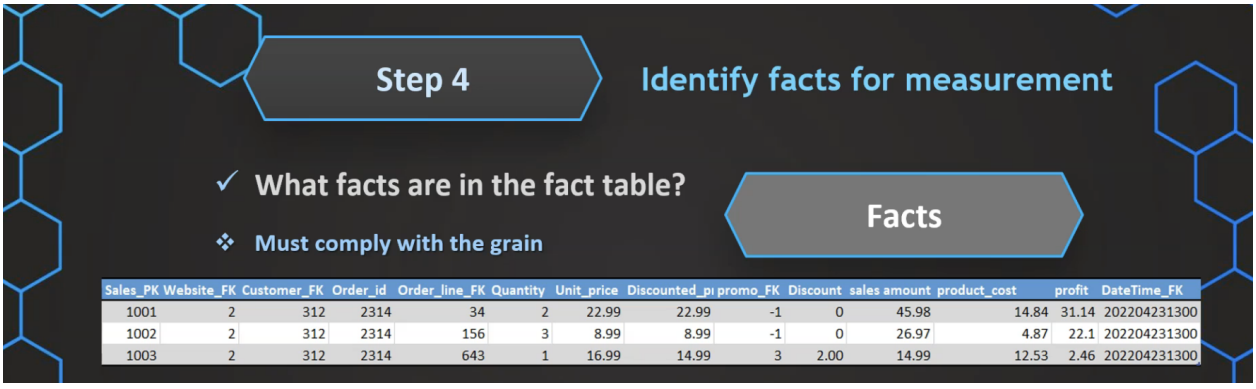
- And with that, we can already see that this is how our data could look like.
- We see have introduced a primary key because there was no real primary key available that is really uniquely identifying every single row. And therefore, we

can just add this basically running number, and this is now just an integer number and this is now uniquely identifying every single row.

- We have the website and note that we have the suffix foreign key. So this is now just a number. And we also did not have to replace any of these IDs because they were already all in these integer numbers and therefore, we could keep them.
- But we have just renamed them to foreign key so that we can see that there are now dimensions associated to those IDs.
- The same thing we have also with the DateTime foreign key. We have just created this key by replacing the date with this longer integer number.
- So we can see this is 2022, and then we have April 23rd. And in here, we have just now rounded the time. So everything fits into one dimension table. So one DateTime dimension.
- It would be also possible to separate the date from the time. So we would have two dimensions. So we have one date and one time dimension. In our case, we have decided to just round this a little bit and then use one dimension table.
- So these are the dimensions that we have now identified. And now in the last step, we need to also identify the facts.

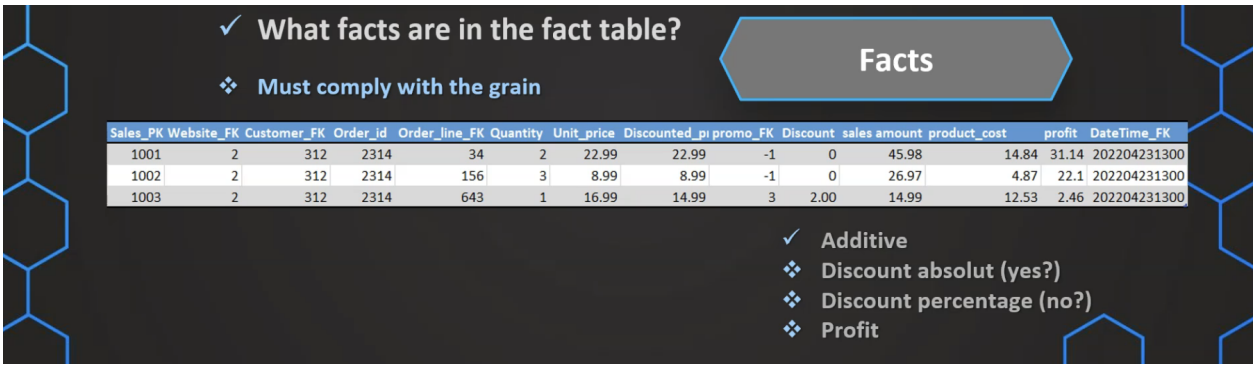
Case study: Identify the facts

- Let's have a look at the last step in the design of our fact table. And this is to identify the facts.



- So what are the facts that are relevant in our fact table? So what are the relevant measurements for the performance that we want to analyze? So this is, of course, now something that needs to comply with the grain.
- If we have some metrics or some measurements that are not complying with the grain but we think we really do need those measurements, then we can go back and think again about the grain, so we can again start at step one.
- But, usually, we can now, in our case, see that all of the measurements are now complying with the grain.
- So for example, we can see the sales amount is just the sales amount in that given row. So this is complying basically with the grain.
- And in this step, we can now decide to remove some facts that we don't need or we can also add additional facts.
- So we can derive some additional calculations from the already existing facts.
- But what we should always keep in mind and consider is if the facts that we are using, if they are additive.
- Because we have learned that additivity is a very important feature in the fact table because this will help us to really use the data well, to aggregate it across all of the different dimensions and get the highest analytical value out of it.
- And in our case, we can see that most of these values are indeed additive, so the sales amount, the product costs.
- We have to be a little bit careful with the unit price and the discounted price, but we can still decide to keep them.
- So in this case, we can keep the unit price but the business users just have to be aware that if they want to calculate total sales amount they can use that value of the sales amount and add those values up. So, this is something that we can do.
- Additionally, we can, as I've mentioned, derive other facts.
- So, in our case, it could be the absolute discount value. So in that case, for example, we have added that already. So the discount is $2.

- In that case, we can just subtract the discount from the unit price. So in that case, we have an absolute discount of $2. And now if this is also already multiplied with the quantity, this is very easily additive and is therefore also a nice fact that can help us with the analysis.
- Something that we should rather avoid is the percentage discount in our case.
- Because we remember percentages and ratios are usually non additive and therefore this is something that should rather be calculated in the BI tool that is used, for example, power BI tableau.



- And this can be done there usually very easily. And therefore we should not physically store this in our Data Warehouse.
- But something that we can do as well is the profit. So we can easily calculate the profit by just subtracting the product cost from the sales amount. Of course, we have to, again, multiply the product cost with the quantity, and then we can subtract the result from the sales amount to end up with the profit.
- Again, it could be, potentially, also calculated in the BI tool. But in that case, we see that the result is really a nice additive value, and therefore to eliminate the risk of some wrong calculations by users or some different ways of calculating this value, we can just physically store the profit so everyone has, basically, the same number no matter what user is using that.
- And, therefore, this is something that can really help to reduce that risk of some wrong numbers, some wrong calculations, and everyone has the same numbers.
- So, this is the final step of our definition and therefore we want to now have a look at the final result of our fact table and the associated dimensions.

- We see that this is our fact table and we have associated with that the website dimension, we have the customer dimension, and also, of course, the products associated to the order line, and then we have also this data dimension.
- And then we have identified a few facts that we can all calculate. So these are the quantity, unit price, discounts, and also sales amount, product cost, and the profit.
- So this would be now the final layout and design of our fact table with the associated facts and the associated dimensions.