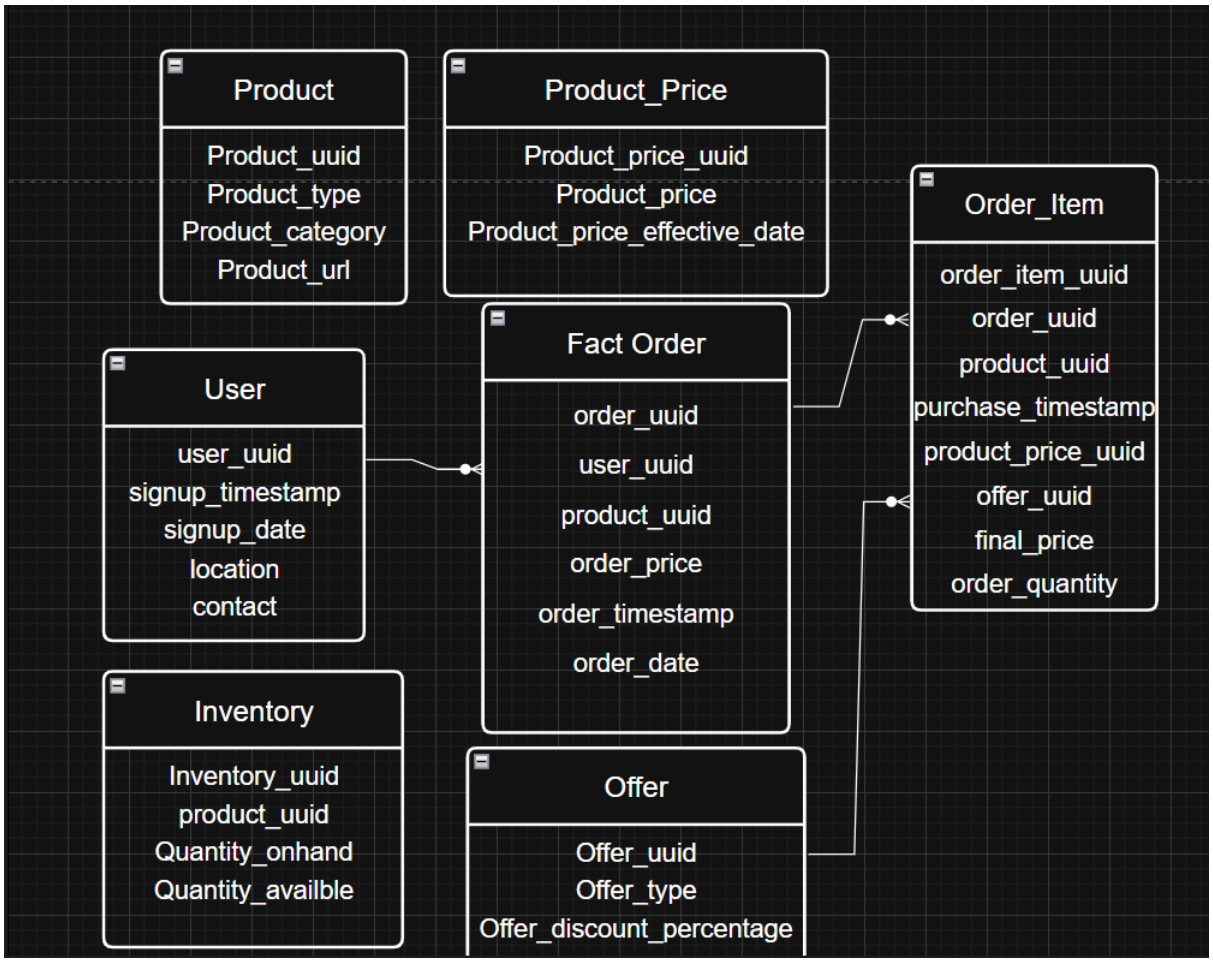
Amazon Data Warehouse Design

Design:



What are the Business use cases to design the DWH for AMAZON?

For the business point of view, we're looking at **customer enrollment**, we want to understand how our **customers acquisition** from a new customers point of view,

How active our customers are which can be translated into how many purchases they make in the first week of acquisition.so are they really engaging with the platform or with the products, is that type of analytics we are looking at !!

What would be the impact of this??

1. We want to link this to our marketing efforts that we are spending as we are spending quite a big amount in terms of marketing promotions & discounts and which one has provided us the most benefit which we will understand form customer acquisition data and how much they are spending

2. In terms of engagement point of view and satisfaction, as they enrolled in the platform may be for a reason becoz a promotion is going on and after enrolling the platform they are not purchasing anything ,not actively engaging then there might be a risk that we might lose this customer down the line and they are not really interested in the platform, we want to make sure we improve the user satisfaction

Who are our users? Will it be used only for analysis purposes ??

This would be primarily for data analysts,ML engineers , this results prove as a decision point for the executives and the leadership to decide around in terms of whether can we continue investing in certain areas, how is our customer acquisition going or what is the trend, going downwards or upwards, all those different things

As per the above criteria, my Data model would look like star schema where the fact will sit at the centre essentially means my “Order” is the base, where my pure focus is generating profits

So my orders would sit at the centre, around this what products I am selling, who are the users essentially the customers who are buying those products

I would like to add inventory and warehouse as well becoz we might not see use cases right now but in future when it comes to order and profit analysis, I see inventory and warehouse management is key role when it comes to purchase becoz amazon might not be the company who creates products but at minor level but they do manage inventory.

My fact will be orders and dimensions will be product, users and essentially may be warehouse will be kept in inventory where I will put it as a fact becoz I will maintain what is the availability so as to how much product is already purchased,The whole idea is if customer wants to purchase a product it has to be available at the inventory for sure.

Do u see any way related to our question/requirements???

No, not directly becoz the question is primarily focused on customer acquisition vs purchase in the first week but the product has to be available in the inventory to be able to purchased.Thats why felt like important to mention as a dimension though it is not directly related to our question

Suppose rarity of the cases like if the customer is ordering something 2-3 times and none of the products are there in the inventory, it doesn’t look good for a customer who just started using a platform, a kind of negative experience

How did we come up with the fact table orders?

When it comes to orders, we want to see that how many products u purchased and what was the amount that u purchased, it will also give analysis on the customer may be that is not part of the question but it gives you high level overview of how many products are you purchasing like what kind of customer you are, are you purchasing the product wit higher or lower value.

Fact order:

1. Order_uuid
2. User_uuid
3. Product_uuid
4. Order_price
5. Order_timestamp
6. Order_date

Now this above ‘order’ attributes gives me high level overview of orders,

Lets not put the product here becoz I want to create another fact table which will have one to many mapping

An order can be divided into multiple suborders for example, each order can have 10 order items and I want to add that separately. So that’s why I removed product from here and I will be adding to order_item

We have order_ timestamp and order_date, was it intentional and why??

Actually I want the one or the other, it won’t be required in all the cases, I don’t want to keep on changing the fields like updating my fields.so when I want my order date I don’t to keep updating my queries to get date from timestamp.

I think it is a redundancy by keeping ??

The person who is changing the order time and playing around with time stamp field,extremely not imp what time zone that their machine is in. There is a very high possibility that the time zone that our table is stored in, they might not be in same time zone and keep changing the dates and then the metrics will be wired.

How are we handling time zone differences in this table(order)??

I am handling time zone differences at my end, I am providing the timestamp and date, so u don’t do ur calculation for ur query on ur side.so it will be same time zone

for everyone, in this case it is UTC timestamp and UTC date, if ur system is in Indian timezone, u don’t see in Indian time until u change

So u r joining two facts i.e. fact order and order_item, which may blow up?

I always want to keep fact order at different grain and fact order_item at different grain.

For example, if a person wants to look at the grain for a given customer, so what is the overall purchase order of that customer which is our use case, but it would also want to drill down that each customer in their orders is buying how many different types of products.

So if I have one table fact i.e order_item and not having fact order separately ,it might blow up the use cases for order and order item, so the users of order item only face the problem of high large data as supposed to not all the users who only want to see the order.so in this case I am having accumulated order price but in order item we don’t need relation on order item

Becoz it’s a fact and it already happened, u didn’t want to update an order price that was happened before, so makes sense to store an accumulator order price, so its an aggregated fact order at customer order level, not drilling much into it ,if u want more details, u go into order items for that order id.Then u see what all the individual order item but from our use case point of view, we rarely may need to do that becoz we are looking at how many orders or may be the value of those orders as a whole.

Order Item:

Order_item_uuid

Order_uuid (one(fact order) to many(order item))

Product_uuid

Purchase_timestamp (we put here becoz someone is looking at order item level, I don’t want to force this table to join with order_timestamp)

Product_price_uuid(to maintain the product price as different table becoz the product price could be changing over a period of time becoz of inflation rates etc any reason, to maintain a track of ur product pricing)

Offer_uuid(what was the actual item price, what was the offer applied and the final price)

Final_price

Order_quantity

USER: (dimension table)

1. User_uuid (customer_uuid) (one to many(fact order))
2. Signup_timestamp
3. Signup_date
4. Location
5. Contact

Offer:

1. Offer_uuid (one to many(order item))

(we need this is we just want to track the percentage of discount that we are offering like when u see on prime day or any festivals provide offer discounts to all the products of similar kind and same percentage. I would like to maintain that track in offer table instead of always updating all key fields and becomes difficult to track.one offer_uuid keep me track the offer which was applied to all these kinds of products)

Lets say u have 10% offer on every product rarely especially in amazon, it would be applicable to a specific type or specifically applicable for first time customers or it could be ur first purchase promotion, etc how are you going for that one?

For example if I want to give 10% discount to all new customers ,then I have a mapping of that in offer , basically offer type kind of I want new customers.so easier to maintain if I have an offer uuid instead of just 10% written here in offer amount

2. Offer_type
3. Offer_discount_percentage

Product:

Product_uuid

Product_type

Product_category

Category_url

Product_price: (fact)

It has the information at a given day, what was my product price,what was the customer charged with offer applied immediately which will be mapped to order item which gives us understanding

We have info about product price on a particular day, whenever there is a change in the product price, there will be entry in this table.

The change can happen either manual update to a product pricing or it could be due to an offer that is applied 10% discount to all products lets say, that might also introduce a change.do we store that type of changes as well??

For that I introduced a separate offer table for that , I want to keep product price and offer separate becoz the product prices pricing is maintained by the company itself but offers that we give to the customers

So this product prize is a base price ,on top of that u can have offers or any type of discounts that u apply, that will be served from the offer table otherwise this product price can quickly blow up lets say we give 5% offer to all products in amazon and essentially create an entire copy of amazon repository again in this product price applying the 10% or 5% price just not ideal

1. Product_price_uuid
2. Product_price
3. Product_price_effective_date

Inventory:

Inventory_uuid

Product_uuid

Quantity_on_hand

Quantity_available

In my inventory there are 10,000 products but all of those have been purchased, I want to keep how much is still available for example customer has asked for product uuid 1 and he wants 10 but I have only 5 available with me, so u can make a decision based on the availability of the product

User query to get the information i.e customer acquisition and purchase in the first week itself, siging up vs product bought after signing up in the first week

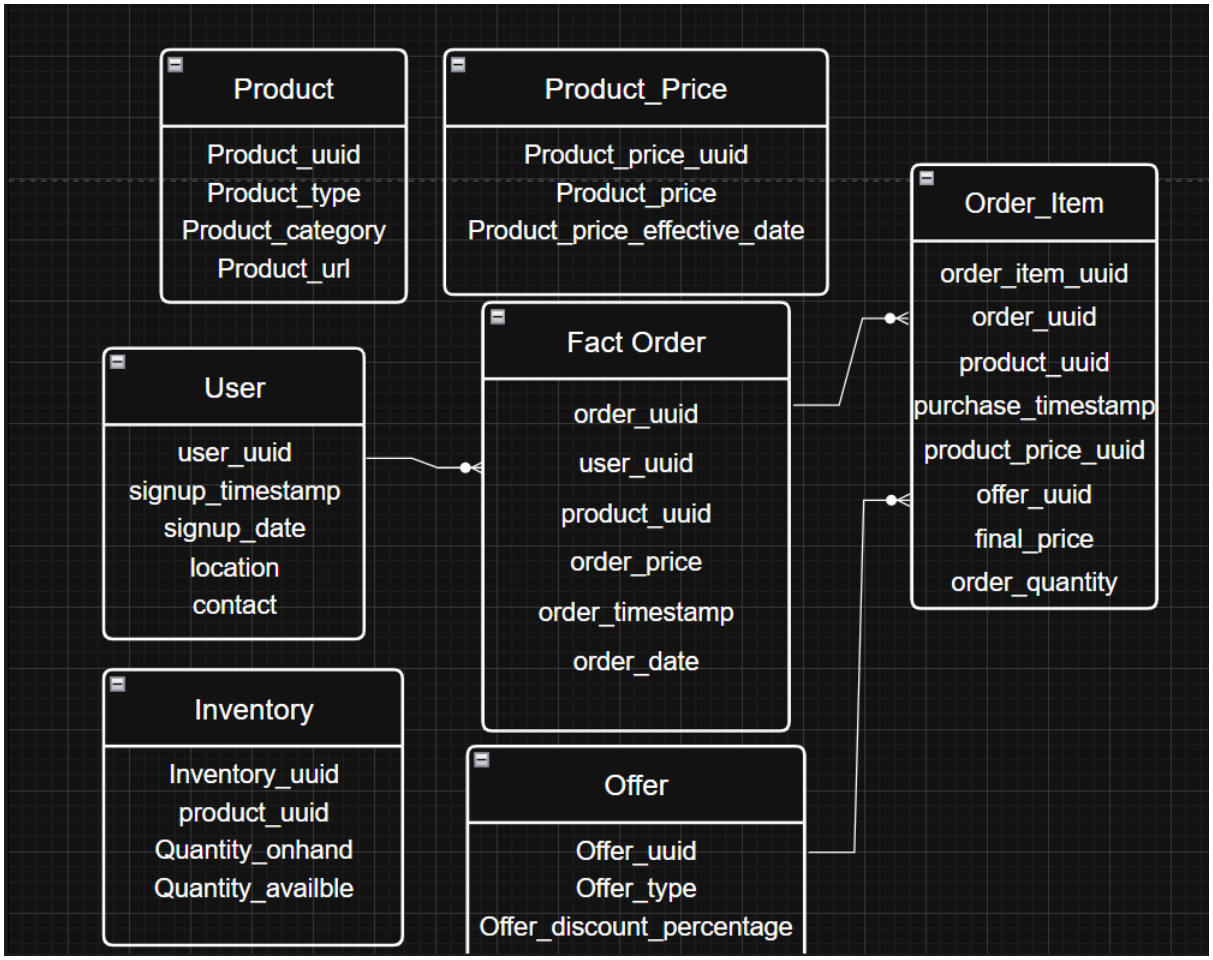
```
With signupweek(Select user_uuid, date_trunc(week,signuptimestamp), signuptimestamp
 signup_week from user where
signuptimestamp> current_date - signuptimestamp- interval'8 weeks'
)
First_week_purchases as(
Select user_uuid, sum(order_price) as purchase_order,count(*) as
count_orders,signup_week  from orders join
Signupweek on orders.user_uuid=Signupweek.user_uuid
Where orders.order_timestamp>= signuptimestamp and orders.order_timestamp
<=signuptimestamp+ interval'7days'
Group by user_uuid

)
Select
week, sum(purchase_order) total_amount_purchases,
sum(count_orders) total_order_count,
sum(purchase_order)/count( distinct user_uuid) as
avg_purchase_order_per_user_per_week,
count(distinct user_uuid) as total_customer_count

From
First_week_purchases
```

```
count(distinct user_uuid) as total_customer_count

From
First_week_purchases
Group by week
```


Product
Product_uuid
Product_type
Product_category
Product_url
Product_Price
Product_price_uuid
Product_price
Product_price_effective_date
Order_Item
order_item_uuid
order_uuid
product_uuid
purchase_timestamp
product_price_uuid
offer_uuid
final_price
order_quantity
Fact Order
order_uuid
user_uuid
product_uuid
order_price
order_timestamp
order_date
User
user_uuid
signup_timestamp
signup_date
location
contact
Inventory
Inventory_uuid
product_uuid
Quantity_onhand
Quantity_availble
Offer
Offer_uuid
Offer_type
Offer_discount_percentage