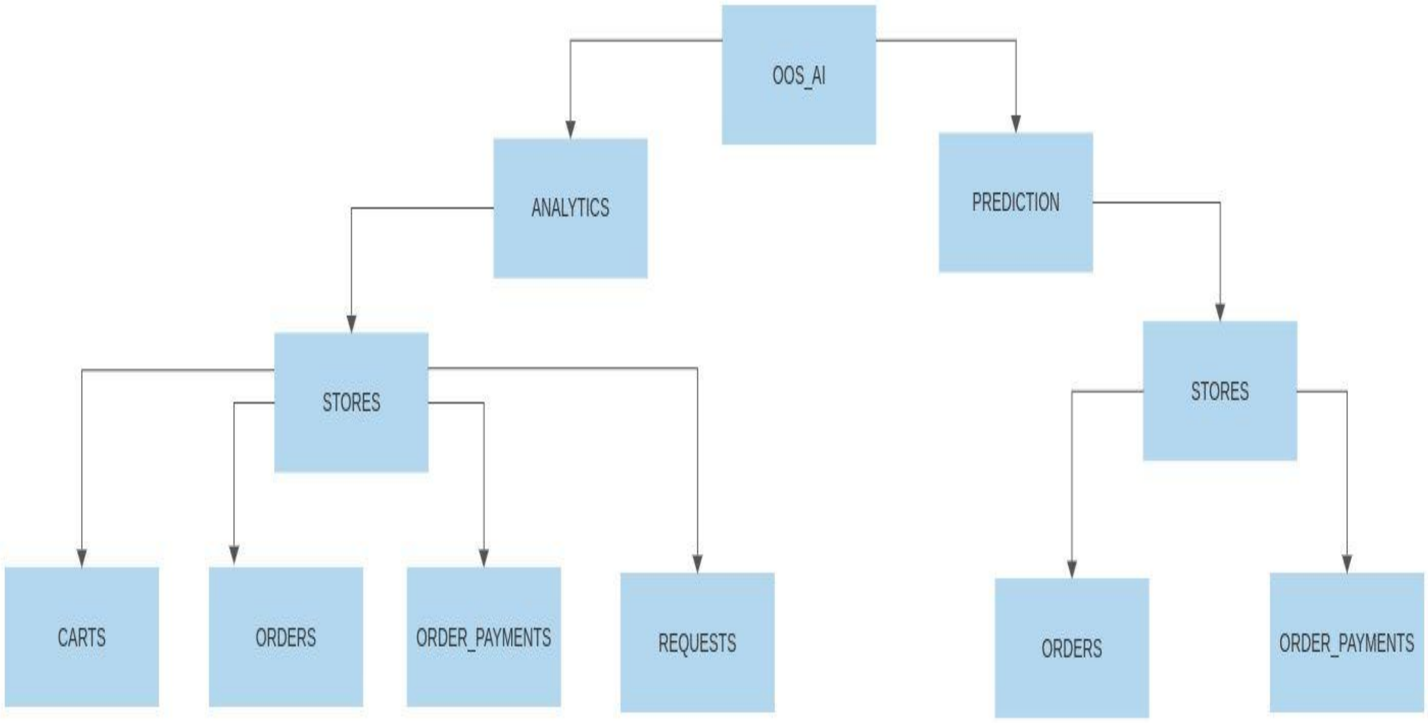
OOS_AI

##Flow of the Procedure ##



API with Parameters

1. /tables - This api is used to display three tables i.e. Most Selling items, Least Selling Items and Most Earning items bound with specific time. Output indicate items + profit + selling_count (most + least)
   a. Parameters:
      i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
      ii. server_token : Store's Server token. Datatype=string.
      iii. start_date : "yyyy-mm-dd".
      iv. end_date : "yyyy-mm-dd".

2. /CustomerRev+Rec - It displays two tables i.e. customer revenue which specify from which customer store receives the highest earning and Recursive customer which display which user order most. Expected Output = {user_name : revenue} + {user_name: order_count} (time bound)
    a. Parameters:
        i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
        ii. server_token : Store's Server token. Datatype=string.
        iii. start_date : "yyyy-mm-dd".
        iv. end_date : "yyyy-mm-dd".

3. /BusyHour - It shows the day + hours and orders at that hour of the day for e.g friday-10 : 3 that means friday at 10 order count is 3. Output = dayname + hour + order_count (time bound)
    a. Parameters:
        i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
        ii. server_token : Store's Server token. Datatype=string.
        iii. start_date : "yyyy-mm-dd".
        iv. end_date : "yyyy-mm-dd".

4. /EarningReport - It shows the revenue (total) date wise. Output shows date + order total (revenue) (timebound)
    a. Parameters:
        i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
        ii. server_token : Store's Server token. Datatype=string.
        iii. start_date : "yyyy-mm-dd".
        iv. end_date : "yyyy-mm-dd".

5. /Predict7 - it Predict the revenue(total) for the next seven days. Output shows ds(date) + yhat(predicted revenue) + yhat_lower(lower limit of predicted revenue) + yhat_upper(upper limit of predicted revenue)
    a. Parameters:
        i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
        ii. server_token : Store's Server token. Datatype=string.

6. /Predict7_Order - It displays the 7 day prediction of order count. Output display future(date) and forecast(order_count).
    a. Parameters:
        i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
        ii. server_token : Store's Server token. Datatype=string.

7. /Busiest_Day_Week - from a given time it shows the day with order count. Output shows the date(dayname) + order_id(order_count). (time bound)
    a. Parameters:
        i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
        ii. server_token : Store's Server token. Datatype=string.
        iii. start_date : "yyyy-mm-dd".
        iv. end_date : "yyyy-mm-dd".

8. /Busiest_Hour - from a given time which hour(0-23) has a high number of orders. Output display date(hour) + order_id(order_count). (time bound)
    a. Parameters:
        i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
        ii. server_token : Store's Server token. Datatype=string.
        iii. start_date : "yyyy-mm-dd".
        iv. end_date : "yyyy-mm-dd".

9. /Time_Analysis - analysis from order it shows the avg time to complete complete order for e.g. waiting for store to accept (order_status-1) , order is accepted by store(status-3),store preparing order(order_status-5) and order_ready(order_status-7) now it display the avg time difference between order status 1&3 and so on. Output has title like "diff1_3" and avg time (time bound)
    a. Parameters:
        i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
        ii. server_token : Store's Server token. Datatype=string.
        iii. start_date : "yyyy-mm-dd".
        iv. end_date : "yyyy-mm-dd".

10. /Time_Analysis_Deliveries - same as time analysis but instead of order status it shows the delivery status analysis. Output has title like "diff1_3" and avg time (time bound)

    a. Parameters:
        i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
        ii. server_token : Store's Server token. Datatype=string.
        iii. start_date : "yyyy-mm-dd".
        iv. end_date : "yyyy-mm-dd".

11. /Monthly_Active_User - We can get the monthly active customers by counting unique user_ids.it shows yearmonth and user count. Output display yearmonth + use_id count (time bound)

a. Parameters:
   i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
   ii. server_token : Store’s Server token. Datatype=string.
   iii. start_date : “yyyy-mm-dd”.
   iv. end_date : “yyyy-mm-dd”.

12. /New_User_Ratio - First we should define what is a new customer. In our dataset, we can assume a new customer is whoever did his/her first purchase in the time window we defined. We will do it monthly. Output shows the yearmonth + user_type + total.(time bound)
    a. Parameters:
       i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
       ii. server_token : Store’s Server token. Datatype=string.
       iii. start_date : “yyyy-mm-dd”.
       iv. end_date : “yyyy-mm-dd”.

13. /User_Monthly_Retention - it indicates how sticky is your service and how well your product fits the market. For making the Monthly Retention Rate visualized, we need to calculate how many customers retained from the previous month. Output indicate the user_id + yearmonth + retention rate (0: not purchased earlier & 1 : purchased earlier) (time_bound).
    a. Parameters:
       i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
       ii. server_token : Store’s Server token. Datatype=string.
       iii. start_date : “yyyy-mm-dd”.
       iv. end_date : “yyyy-mm-dd”.

14. /User_Weekly_Retention - same as User_monthly_retention but instead of showing month wise it shows week wise. Output indicate the week_number + user_id + retention rate (0: not purchased earlier & 1 : purchased earlier) (time_bound)
    a. Parameters:
       i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
       ii. server_token : Store’s Server token. Datatype=string.
       iii. start_date : “yyyy-mm-dd”.
       iv. end_date : “yyyy-mm-dd”.

15. /Customer_Segment - decide the customer based on RFM(RFM stands for Recency - Frequency - Monetary Value). Theoretically we will have segments like below: 1.Low Value: Customers who are less active than others, not very frequent buyers and generates very low - zero - maybe negative revenue. 2.Mid Value: In the middle of

everything. Often using our platform (but not as much as our High Values), fairly frequent and generates moderate revenue. 3.High Value: The group we don't want to lose. High Revenue, Frequency and low Inactivity. (time_bound)

Recency - how recently users ordered.
Frequency - how frequently users ordered.
Monetary Value - which user gives high revenue(order_total).

Output shows the user_id + Recency (days user last purchase) + RecencyCluster (0:worst to 3:best) + Frequency (order count) + FrequencyCluster (0:worst to 3:best) + revenue(total) + RevenueCluster(0:worst to 3:best) + Segment

a. Parameters:
   i. _id : specify the store_id inorder to identify for which store we have to perform the according operation. Datatype=string.
   ii. server_token : Store's Server token. Datatype=string.
   iii. start_date : "yyyy-mm-dd".
   iv. end_date : "yyyy-mm-dd".