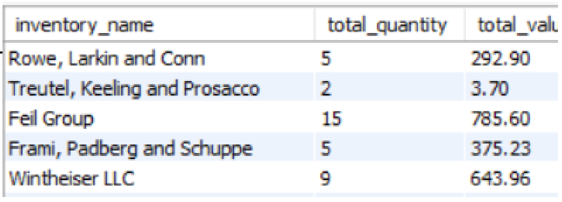
IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:

1) **Calculate the total quantity and value of products sold by a particular supplier in a the Month of May?**

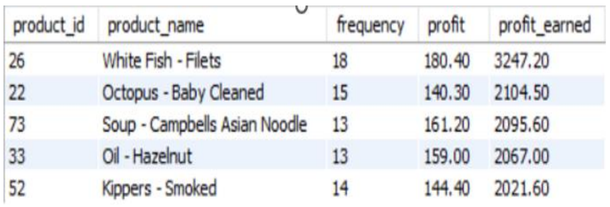
SELECT s.inventory_name, SUM(po.quantity_ordered) as total_
SUM(po.quantity_ordered * po.selling_price) as total_value
FROM orders po JOIN inventory s ON po.inventory_id
= s.inventory_id WHERE left(order_date, 7) = '2022-05'
GROUP BY s.inventory_name;



inventory_name	total_quantity	total_valu
Rowe, Larkin and Conn	5	292.90
Treutel, Keeling and Prosacco	2	3.70
Feil Group	15	785.60
Frami, Padberg and Schuppe	5	375.23
Wintheiser LLC	9	643.96

2) **Retrieve the Products that gained the maximum profits for the Company**

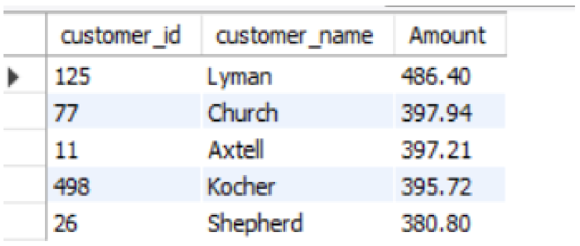
SELECT p.product_id,p.product_name, a.frequency, (p.selling_price-p.actual_price) as profit, (frequency*(p.selling_price-p.actual_price)) as profit_earned from products as p right join (SELECT product_id, count(product_id) as Frequency FROM orders GROUP BY product_id) as a on a.product_id=p.product_id order by profit_earned desc limit 5;



product_id	product_name	frequency	profit	profit_earned
26	White Fish - Filets	18	180.40	3247.20
22	Octopus - Baby Cleaned	15	140.30	2104.50
73	Soup - Campbells Asian Noodle	13	161.20	2095.60
33	Oil - Hazelnut	13	159.00	2067.00
52	Kippers - Smoked	14	144.40	2021.60

3) **Find the TOP 5 Customers who purchased more number of items?**

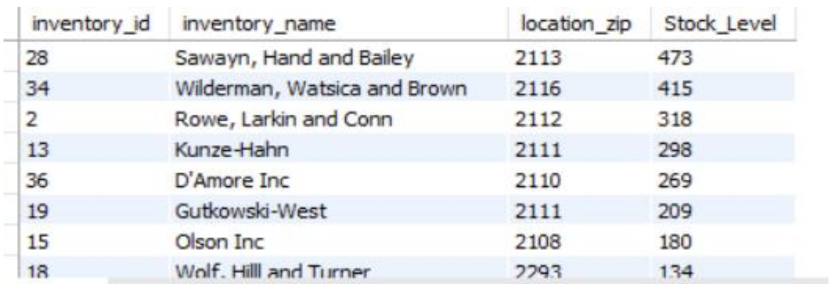
select c.customer_id, ce.customer_name,
sum(selling_price) as Amount from customer as c
join orders as co on c.order_id=co.order_id
join customers as ce on c.customer_id=ce.customer_id
group by c.customer_id
order by Amount desc limit 5;



customer_id	customer_name	Amount
125	Lyman	486.40
77	Church	397.94
11	Axtell	397.21
498	Kocher	395.72
26	Shepherd	380.80

4) **List all the Inventories Based on the Boston Location with the Stock mounted in their Storage?**

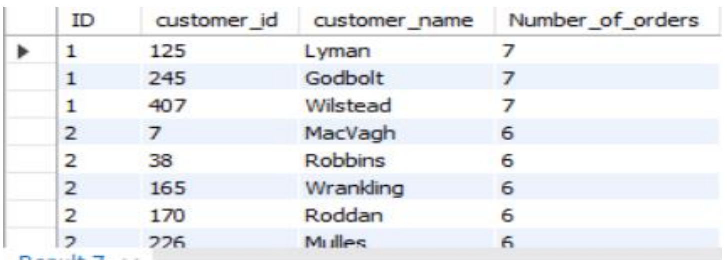
SELECT i.inventory_id, i.inventory_name , l.location_zip, sum(ise.quantity) as Stock_Level from inventory as i join locations as l
on i.location_id = l.location_id
join inventory_stocks as ise
on i.inventory_id = ise.inventory_id
where l.location_name = "Boston"
group by ise.inventory_id
order by Stock_level desc;



inventory_id	inventory_name	location_zip	Stock_Level
28	Sawayn, Hand and Bailey	2113	473
34	Wilderman, Watsica and Brown	2116	415
2	Rowe, Larkin and Conn	2112	318
13	Kunze-Hahn	2111	298
36	D'Amore Inc	2110	269
19	Gutkowski-West	2111	209
15	Olson Inc	2108	180
18	Wolf, Hill and Turner	2293	134

5) **Rank the customers in order that are the most regular to the business and have placed most Orders?**

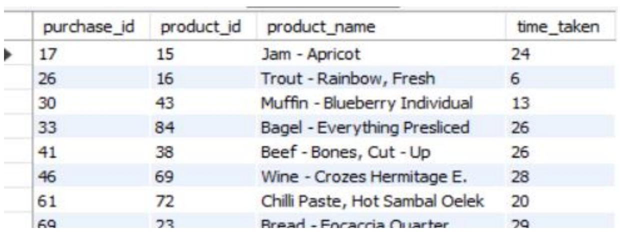
select dense_rank() over (order by count(co.order_id) desc) as ID,
c.customer_id,c.customer_name,count(co.order_id)
as Number_of_orders from customers as c
join customer_orders as co
on c.customer_id=co.customer_id
group by customer_id
order by Number_of_orders desc;



ID	customer_id	customer_name	Number_of_orders
1	125	Lyman	7
1	245	Godbolt	7
1	407	Wilstead	7
2	7	MacVagh	6
2	38	Robbins	6
2	165	Wrankling	6
2	170	Roddan	6
2	226	Mulles	6

6) List all the orders that arived from the Supplier to the inventory within 30 days from ordering?

with date_diff_cte as
(select purchase_id, product_id, datediff(delivery_date, order_date) as time_taken from purchase_orders)
select df.purchase_id, df.product_id,p.product_name, df.time_taken from date_diff_cte df, products p where df.product_id=p.product_id and df.time_taken < 30;



purchase_id	product_id	product_name	time_taken
17	15	Jam - Apricot	24
26	16	Trout - Rainbow, Fresh	6
30	43	Muffin - Blueberry Individual	13
33	84	Bagel - Everything Presliced	26
41	38	Beef - Bones, Cut - Up	26
46	69	Wine - Crozes Hermitage E.	28
61	72	Chilli Paste, Hot Sambal Oelek	20
69	23	Bread - Focaccia Quarter	29

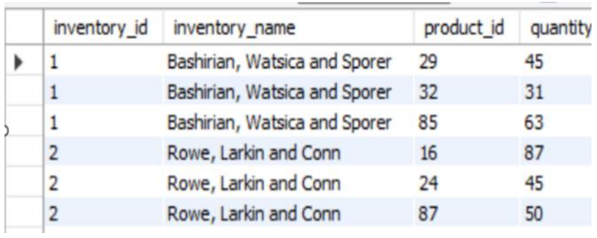
7) Create a Stored Procedure that can add values to Table when order is placed and optimize the rows?

delimiter $$
drop procedure if exists place_orders;
create procedure place_orders()
begin
declare v_order_id int; declare v_inventory_id int; declare v_quantity_ordered int; declare v_product_id int; declare v_selling_price int;
select count(*) as order_id, inventory_id,quantity_ordered, product_id, selling_price into v_order_id,v_inventory_id, v_quantity_ordered, v_product_id, v_selling_price
from orders where order_id=1;
insert into orders values (v_order_id+10002, v_inventory_id, now(), v_product_id, v_selling_price, v_quantity_ordered);
select "Order Placed";
end $$

call place_orders;

8) Write Query Results the Stocks of products which having more than the average stock in that inventories?

select i.inventory_id,i.inventory_name,iss.product_id, iss.quantity from inventory as i join inventory_stocks as iss where iss.inventory_id=i.inventory_id and iss.quantity > (select avg(quantity) from inventory_stocks as iss2 where iss2.inventory_id=iss.inventory_id);



inventory_id	inventory_name	product_id	quantity
1	Bashirian, Watsica and Sporer	29	45
1	Bashirian, Watsica and Sporer	32	31
1	Bashirian, Watsica and Sporer	85	63
2	Rowe, Larkin and Conn	16	87
2	Rowe, Larkin and Conn	24	45
2	Rowe, Larkin and Conn	87	50