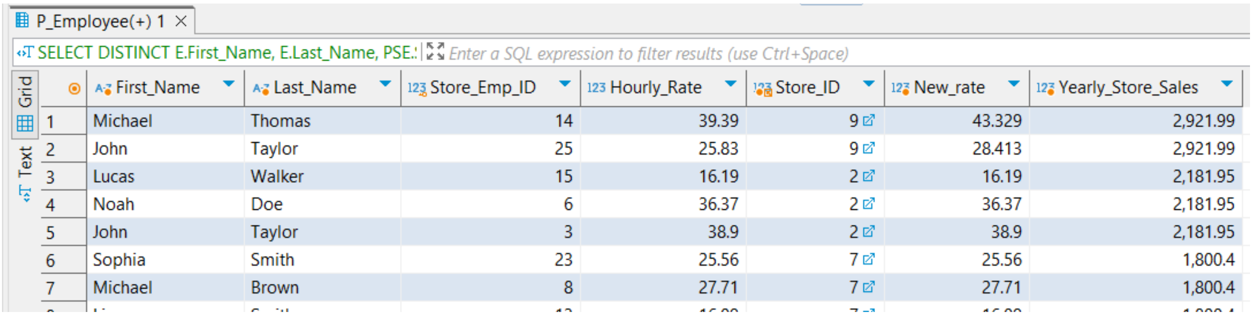
Report 1: Reward employees who work for the store with the highest yearly sales.

Benefits & Business Uses (Value):
This query first looks at the yearly sales for the different stores of our shoe retail business. From the yearly sales it takes the store with the highest yearly sales and gives all employees who belong to that store a 10% raise in their hourly rate. This query rewards employees whose hard work and dedication resulted in their store having the highest yearly sales figures. This also provides motivation for store employees and encourages them to work as a team.

Assumptions:

- We are assuming that our stores have similar numbers for employees, store selection, and store size with only the sales variable changing across stores.

SQL Query:
```sql
SELECT DISTINCT E.First_Name, E.Last_Name, PSE.Store_Emp_ID, PSE.Hourly_Rate,
PES.Store_ID,
(CASE
WHEN PES.Store_ID = (
SELECT Store_ID
FROM P_Sales
WHERE DATEDIFF(NOW(), Sale_Date) / 365 < 1
GROUP BY Store_ID
ORDER BY SUM(Total_Amount) DESC
LIMIT 1
) THEN PSE.Hourly_Rate * 1.1
ELSE PSE.Hourly_Rate
END) AS New_rate,
SUM(CASE
WHEN DATEDIFF(NOW(), ps2.Sale_Date) / 365 < 1 THEN ps2.Total_Amount
ELSE 0
END) AS Yearly_Store_Sales
FROM P_Store_Employee PSE
JOIN P_Employee_Shift_Assignment PES ON PSE.Store_Emp_ID = PES.Store_Employee_ID
JOIN P_Store PS ON PES.Store_ID = PS.Store_ID
JOIN P_Employee E ON PSE.Employee_ID = E.Employee_ID
JOIN P_Sales ps2 ON ps2.Store_ID = PS.Store_ID
GROUP BY E.First_Name, E.Last_Name, PSE.Store_Emp_ID, PSE.Hourly_Rate, PES.Store_ID
ORDER BY Yearly_Store_Sales DESC;
```



```
SELECT DISTINCT E.First_Name, E.Last_Name, PSE.:
Enter a SQL expression to filter results (use Ctrl+Space)
```

	First_Name	Last_Name	Store_Emp_ID	Hourly_Rate	Store_ID	New_rate	Yearly_Store_Sales
1	Michael	Thomas	14	39.39	9	43.329	2,921.99
2	John	Taylor	25	25.83	9	28.413	2,921.99
3	Lucas	Walker	15	16.19	2	16.19	2,181.95
4	Noah	Doe	6	36.37	2	36.37	2,181.95
5	John	Taylor	3	38.9	2	38.9	2,181.95
6	Sophia	Smith	23	25.56	7	25.56	1,800.4
7	Michael	Brown	8	27.71	7	27.71	1,800.4

Report 2: Calculate sales metrics and suggest discounts based on the quantity sold.

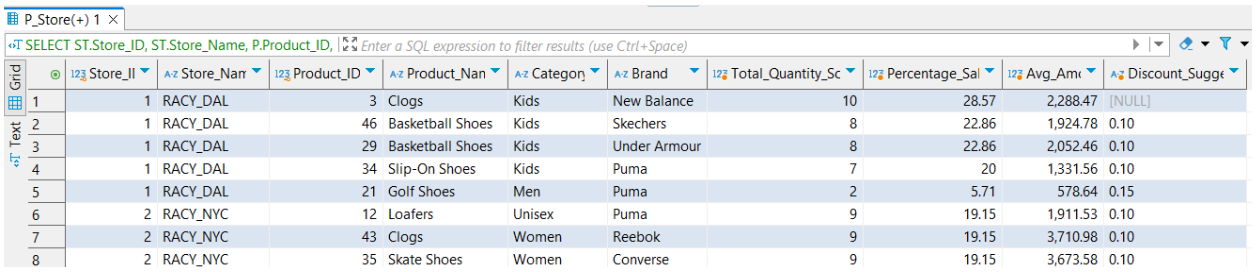
Benefits & Business Uses (Value): This query provides comprehensive insights into product performance at the store level, empowering store managers and decision-makers to identify high and low-performing products. By understanding each product’s contribution to overall sales, stores can make informed decisions about inventory management, promotions, and discounts. The discount suggestion metric is especially valuable for optimizing pricing strategies to boost sales of slower-moving items, enhancing revenue and inventory turnover. Additionally, tracking average spending per product helps tailor marketing strategies and refine product placements to match customer preferences, resulting in improved customer satisfaction and retention, as well as maximizing profitability per square foot of retail space.

Assumptions:

- We are binning the quantities sold as follows: 0 to 3, It will give a 15% promotional discount. 4 to 9, it will give a 10% discount. More than 9, it won’t give any discount.

SQL Query:

```
SELECT ST.Store_ID, ST.Store_Name, P.Product_ID, P.Product_Name, P.Category,
P.Brand,
SUM(S.Qyantity) AS Total_Quantity_Sold, -- Total quantity sold per product per store
ROUND((SUM(S.Qyantity) / (SELECT SUM(Qyantity) FROM P_Sales WHERE Store_ID =
ST.Store_ID) * 100), 2) AS Percentage_Sales, -- Percentage of total sales by each
product in the store
ROUND(AVG(S.Total_Amount), 2) AS Avg_Amount, -- Average amount spent on each product
CASE WHEN SUM(S.Qyantity) BETWEEN 4 AND 9 THEN '0.10'
WHEN SUM(S.Qyantity) <= 3 THEN '0.15' ELSE NULL
END AS Discount_Suggestion -- Discount based on the total quantity sold
FROM P_Sales S
JOIN P_Product P ON S.Product_ID = P.Product_ID
JOIN P_Store ST ON S.Store_ID = ST.Store_ID
GROUP BY ST.Store_ID, ST.Store_Name, P.Product_ID, P.Product_Name, P.Category,
P.Brand
HAVING Total_Quantity_Sold > 0
ORDER BY ST.Store_ID, Total_Quantity_Sold DESC;
```



P_Store(+) 1

SELECT ST.Store_ID, ST.Store_Name, P.Product_ID,

	Store_II	Store_Nam	Product_ID	Product_Nan	Category	Brand	Total_Quantity_Sc	Percentage_Sal	Avg_Amc	Discount_Sugge
1	1	RACY_DAL	3	Clogs	Kids	New Balance	10	28.57	2,288.47	[NULL]
2	1	RACY_DAL	46	Basketball Shoes	Kids	Skechers	8	22.86	1,924.78	0.10
3	1	RACY_DAL	29	Basketball Shoes	Kids	Under Armour	8	22.86	2,052.46	0.10
4	1	RACY_DAL	34	Slip-On Shoes	Kids	Puma	7	20	1,331.56	0.10
5	1	RACY_DAL	21	Golf Shoes	Men	Puma	2	5.71	578.64	0.15
6	2	RACY_NYC	12	Loafers	Unisex	Puma	9	19.15	1,911.53	0.10
7	2	RACY_NYC	43	Clogs	Women	Reebok	9	19.15	3,710.98	0.10
8	2	RACY_NYC	35	Skate Shoes	Women	Converse	9	19.15	3,673.58	0.10

Report 3: Determine updated customer loyalty points and membership level based on purchases.

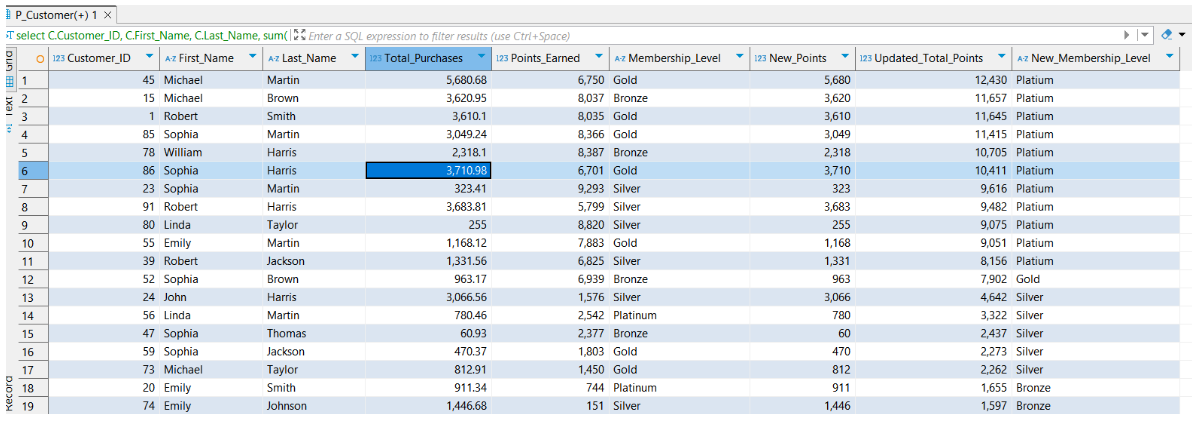
Benefits & Business Uses (Value): It is important that the business is able to see a customer's updated loyalty points and which loyalty tier (membership level) they belong to. Customers in the Gold and Platinum tiers may respond differently to marketing tactics than customers in the Bronze and Silver tiers. In addition, the business may offer different promotional benefits to customers based on loyalty tier. For example, customers may be incentivized to enter into a higher membership level because Platinum members are offered more discounts than Gold members.

Assumptions:

- We are assuming that the loyalty points the customer already has in the DB were not derived from their purchases.
- We want to add 1 point per 1 full dollar of purchases (ex. $49.90 of purchases earns 49 points).

SQL Query:

```
SELECT C.Customer_ID, C.First_Name, C.Last_Name, SUM(S.Total_Amount) AS
Total_Purchases, CL.Points_Earned, CL.Membership_Level,
TRUNCATE(SUM(S.Total_Amount), 0) AS New_Points,
(TRUNCATE(SUM(S.Total_Amount), 0) + CL.Points_Earned) AS Updated_Total_Points,
(CASE WHEN (TRUNCATE(SUM(S.Total_Amount), 0) + CL.Points_Earned) BETWEEN 0 AND 2000
THEN 'Bronze'
WHEN (TRUNCATE(SUM(S.Total_Amount), 0) + CL.Points_Earned) BETWEEN 2001 AND 5000 THEN
'Silver'
WHEN (TRUNCATE(SUM(S.Total_Amount), 0) + CL.Points_Earned) BETWEEN 5001 AND 8000 THEN
'Gold'
WHEN (TRUNCATE(SUM(S.Total_Amount), 0) + CL.Points_Earned) > 8000 THEN 'Platinum'
END) AS New_Membership_Level
FROM P_Sales S
JOIN P_Customer C ON S.Customer_ID = C.Customer_ID
JOIN P_Customer_Loyalty CL ON C.Customer_ID = CL.Customer_ID
GROUP BY C.Customer_ID
ORDER BY Updated_Total_Points DESC;
```



P_Customer(+) 1

select C.Customer_ID, C.First_Name, C.Last_Name, sum(

Enter a SQL expression to filter results (use Ctrl+Space)

	Customer_ID	First_Name	Last_Name	Total_Purchases	Points_Earned	Membership_Level	New_Points	Updated_Total_Points	New_Membership_Level
1	45	Michael	Martin	5,680.68	6,750	Gold	5,680	12,430	Platium
2	15	Michael	Brown	3,620.95	8,037	Bronze	3,620	11,657	Platium
3	1	Robert	Smith	3,610.1	8,035	Gold	3,610	11,645	Platium
4	85	Sophia	Martin	3,049.24	8,366	Gold	3,049	11,415	Platium
5	78	William	Harris	2,318.1	8,387	Bronze	2,318	10,705	Platium
6	86	Sophia	Harris	3,710.98	6,701	Gold	3,710	10,411	Platium
7	23	Sophia	Martin	323.41	9,293	Silver	323	9,616	Platium
8	91	Robert	Harris	3,683.81	5,799	Silver	3,683	9,482	Platium
9	80	Linda	Taylor	255	8,820	Silver	255	9,075	Platium
10	55	Emily	Martin	1,168.12	7,883	Gold	1,168	9,051	Platium
11	39	Robert	Jackson	1,331.56	6,825	Silver	1,331	8,156	Platium
12	52	Sophia	Brown	963.17	6,939	Bronze	963	7,902	Gold
13	24	John	Harris	3,066.56	1,576	Silver	3,066	4,642	Silver
14	56	Linda	Martin	780.46	2,542	Platinum	780	3,322	Silver
15	47	Sophia	Thomas	60.93	2,377	Bronze	60	2,437	Silver
16	59	Sophia	Jackson	470.37	1,803	Gold	470	2,273	Silver
17	73	Michael	Taylor	812.91	1,450	Gold	812	2,262	Silver
18	20	Emily	Smith	911.34	744	Platinum	911	1,655	Bronze
19	74	Emily	Johnson	1,446.68	151	Silver	1,446	1,597	Bronze