

Lawson On-the-Go Database Management Solutions

MANAGEMENT OF ORGANIZATIONAL DATA GROUP PROJECT
SESSION 1- GROUP 2

Poorna K Narasimhan
Prerana Das
Rupal Bilaiya
Sanjana Santhanakrishnan
Souradeep Chakroborty
Sree Kailash Ravichandar

CLIENT BACKGROUND:

The grab-and-go type of food has always been an important demand of boilermakers during weekday classes. Lawson OTG is Purdue's unique premiere On-the-GO dining that offers extensive meal options keeping in mind the fast-paced lives of the students. With their long menu, speedy supply of sandwiches, coffee and much more to students who are running for classes, it is imperative to create an efficient work-flow structure to cater to students' needs and create efficient shift allotment system.

INTRODUCTION - PROJECT OBJECTIVE:

The objective of the study is to –

- Determine the data flow of multiple databases used by Lawson OTG
- Analyze/query the data to find business solutions to the problem statement
- Create an effective inventory management system for OTG

KEY TAKEAWAYS:

- ✓ Resource allocation and rationing is one of the more important aspects for any business to thrive. Managers at Lawson need to investigate the uneven distribution of shifts and restructure the resource allocation keeping in the mind the varied trend in business demands across the week.
- ✓ Inventory management and stock keeping is important for a business that has varied demand every day. Lawson should investigate managing their inventory better as the ingredients for one of their bestselling items – Blooming Breakfast – are out of stock.
- ✓ Lawson needs to make more products available under Meal Swipe as some items are available under Credit card only. As most items are sold at a higher price under Meal swipe, and since most people do not take the add-ons that come alongside, including more items under Meal Swipe will make Lawson more lucrative.

DESCRIPTION OF THE DATASET:

We were provided with 8 tables as hard copies by the Lawson OTG –

1. Employee related information:

1. Employee table has employee information such as name, phone number etc.
2. Role table has salary for each role
3. Employee shift has the shift of each employee with shift timings

2. Food Inventory data –

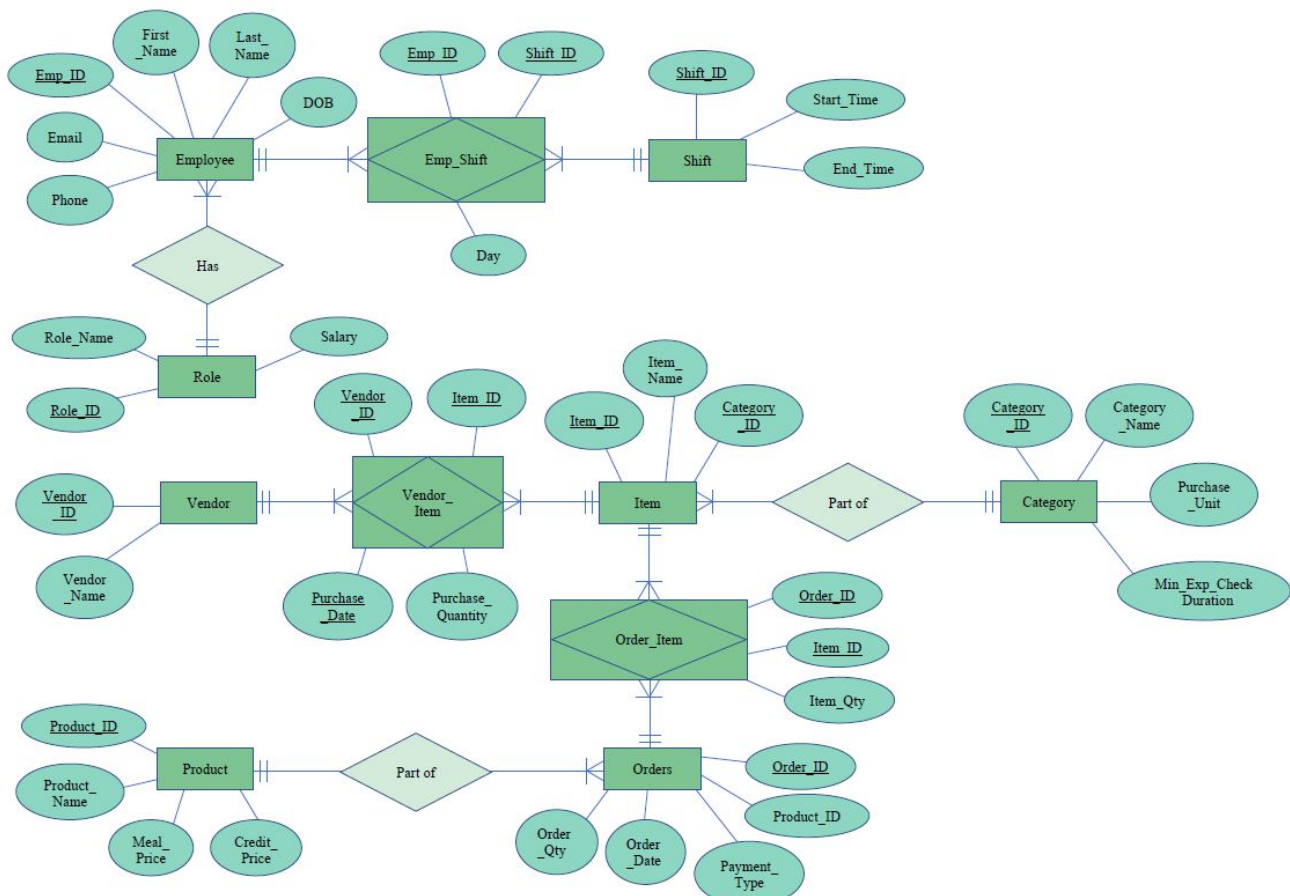
1. Product Item includes Item and category composition
2. Category tables includes category name, purchasing quantity and expiry date
3. Vendor data includes vendor details

3. Food Orders data –

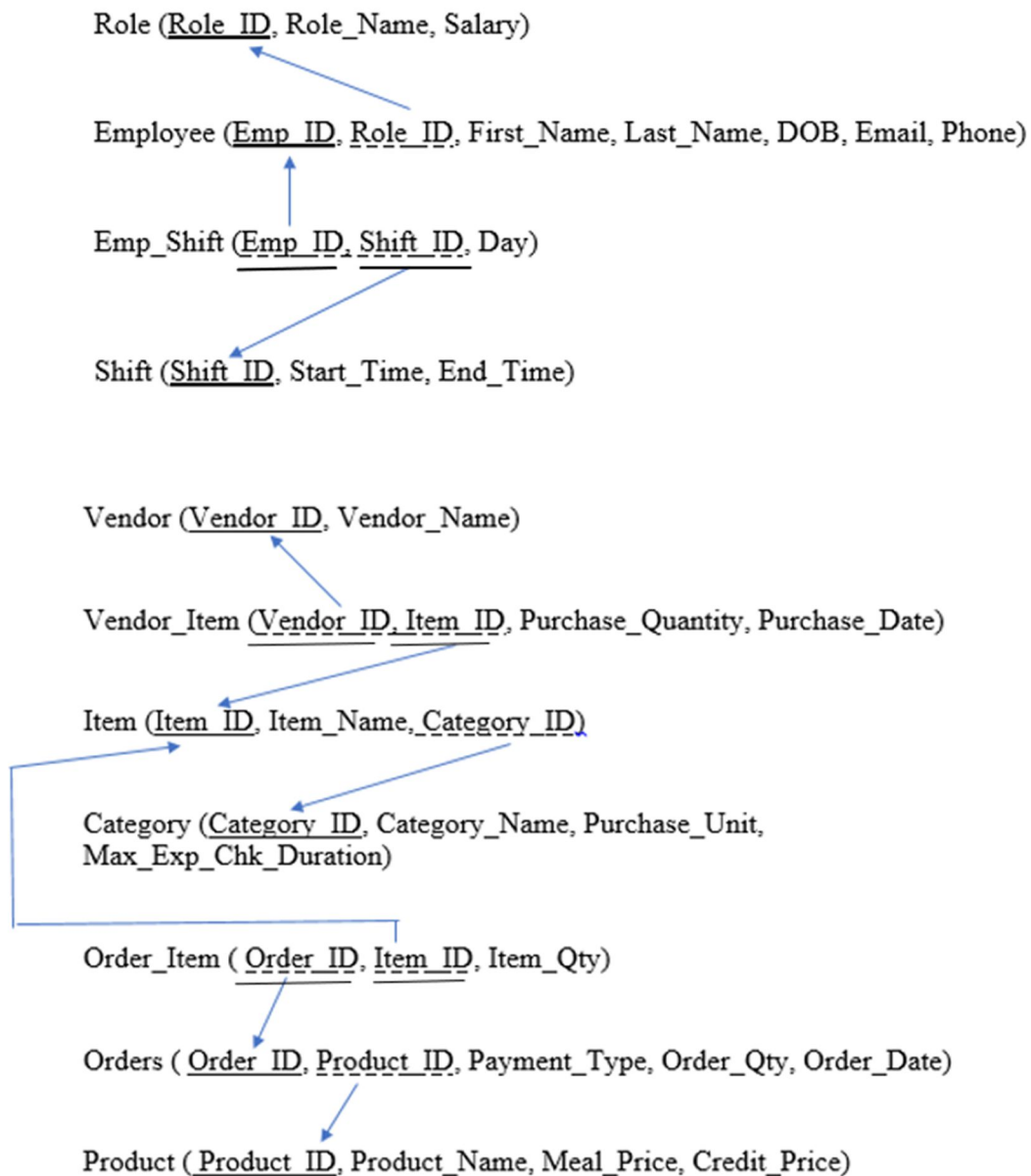
1. Order details which includes Order ID, payment type, dates etc.
2. Product details which includes meal prices of all the products

We have created 8 entity, 3 associative entity and 3 relationships in our dataset.

CONCEPTUAL DATA MODELLING - ERD:



RELATIONAL DATA MODEL



DATA DESIGN

The Employee data obtained was in 2 NF. It had a transitive dependency Role -> Salary.

Emp_ID	First_Name	Last_Name	DOB	Phone	Email	Role	Salary
439	ALICE	SMITH	6/14/1996	+1 765-480-8844	smitha7@	STUDENT SUPERV	12
361	LIAM	WILLIAMS	8/6/2000	+1 224-478-5416	williams13@	STUDENT ASSOCIA	10
550	NOAH	JOHNSON	8/22/1996	+1 650-476-3499	johnsonn2@	STUDENT ASSOCIA	10
457	OLIVER	JONES	3/16/1997	+1 312-478-1597	joneso1@	STUDENT SUPERV	12
455	ELIJAH	BROWN	10/18/1997	+1 847-477-9544	browne0@	STUDENT SUPERV	15
439	OLIVIA	DAVIS	4/24/1996	+1 765-480-8624	daviso9@	STUDENT SUPERV	12
556	EMMA	MILLER	7/21/1999	+1 267-483-5751	millere7@	STUDENT SUPERV	12

So, we have created a new table for Role with Salary details in it.

Emp_ID	First_Nam	Last_Name	DOB	Phone	Email	Role_ID
439	ALICE	SMITH	6/14/1996	+1 765-480-8844	smitha7@	3
361	LIAM	WILLIAMS	8/6/2000	+1 224-478-5416	williams13@	4
550	NOAH	JOHNSON	8/22/1996	+1 650-476-3499	johnsonn2@	4
457	OLIVER	JONES	3/16/1997	+1 312-478-1597	joneso1@	3
455	ELIJAH	BROWN	10/18/1997	+1 847-477-9544	browne0@	2
439	OLIVIA	DAVIS	4/24/1996	+1 765-480-8624	daviso9@	3
556	EMMA	MILLER	7/21/1999	+1 267-483-5751	millere7@	3

Role_ID	Role_Name	Salary
1	STUDENT MANAGER	\$18
2	STUDENT SUPERVISOR	\$15
3	STUDENT SUPERVISOR IN TRAINING	\$18
4	STUDENT ASSOCIATE	\$18

The Orders data obtained was in 0NF as two different prices were mentioned in one column as a composite attribute.

Order_ID	Product_ID	Product_Name	Order_Date	Order_Qty	Payment_Type	Price
1	1	Butterbeer Latte	11/1/2021	5	Meal/Credit	0, 4.69
2	2	Fountain Drink	11/1/2021	4	Meal/Credit	0, 1.89
3	3	Water Cup	11/1/2021	1	Meal	0, 0.25
4	4	Monster Zero Ultra 16 oz	11/1/2021	3	Meal/Credit	0, 2.89
5	5	Minute Maid 20 oz Bottles	11/1/2021	4	Meal/Credit	0, 1.99
6	6	Body Armor Water	11/1/2021	1	Meal	0, 2.29
7	7	Body Armor Flavor	11/1/2021	1	Meal	0, 2.79
8	8	Dasani 20 oz	11/1/2021	1	Meal	0, 1.79

So, we converted this to 1NF by splitting the Price into Meal_Price and Credit_Price. Since there were no partial dependencies, this table was directly in 2NF.

Order_ID	Product_ID	Product_Name	Order_Date	Order_Qty	Payment_Type	Meal_Price	Credit_Price
1	1	Butterbeer Latte	11/1/2021	5	Meal/Credit	0	4.69
2	2	Fountain Drink	11/1/2021	4	Meal/Credit	0	1.89
3	3	Water Cup	11/1/2021	1	Meal	0	0.25
4	4	Monster Zero Ul	11/1/2021	3	Meal/Credit	0	2.89
5	5	Minute Maid 20	11/1/2021	4	Meal/Credit	0	1.99
6	6	Body Armor Wat	11/1/2021	1	Meal	0	2.29
7	7	Body Armor Flav	11/1/2021	1	Meal	0	2.79
8	8	Dasani 20 oz	11/1/2021	1	Meal	0	1.79

This table had transitive dependencies –

Product_ID - > Product_Name, Meal_Price, Credit_Price

And so, we converted this to 3NF by creating a new table called Product with these attributes.

Order_ID	Product_ID	Order_Date	Order_Qty	Payment_Type
1	1	11/1/2021	5	Meal/Credit
2	2	11/1/2021	4	Meal/Credit
3	3	11/1/2021	1	Meal
4	4	11/1/2021	3	Meal/Credit
5	5	11/1/2021	4	Meal/Credit
6	6	11/1/2021	1	Meal
7	7	11/1/2021	1	Meal
8	8	11/1/2021	1	Meal

Product_ID	Product_Name	Meal_Price	Credit_Price
1	Butterbeer Latte	0	4.69
2	Fountain Drink	0	1.89
3	Water Cup	0	0.25
4	Monster Zero Ultra 16 oz	0	2.89
5	Minute Maid 20 oz Bottles	0	1.99
6	Body Armor Water	0	2.29
7	Body Armor Flavor	0	2.79
8	Dasani 20 oz	0	1.79

Similar, Vendor_Item table was in 1NF due to a partial functional dependency –

Vendor_ID -> Vendor_Name

Here Vendor_ID, Item_ID and Purchase_Date are all primary keys.

Vendor_ID	Item_ID	Purchase_Date	Vendor_Name	Purchase_Quantity
8278	1083	10/23/2021	Dean Foods/Schenkel's Dairy	2
8278	1084	10/23/2021	Dean Foods/Schenkel's Dairy	3
8278	1085	10/23/2021	Dean Foods/Schenkel's Dairy	2
8278	1086	10/23/2021	Dean Foods/Schenkel's Dairy	5
8278	1021	10/23/2021	Dean Foods/Schenkel's Dairy	2
8278	1022	10/23/2021	Dean Foods/Schenkel's Dairy	2
8278	1057	10/23/2021	Dean Foods/Schenkel's Dairy	2

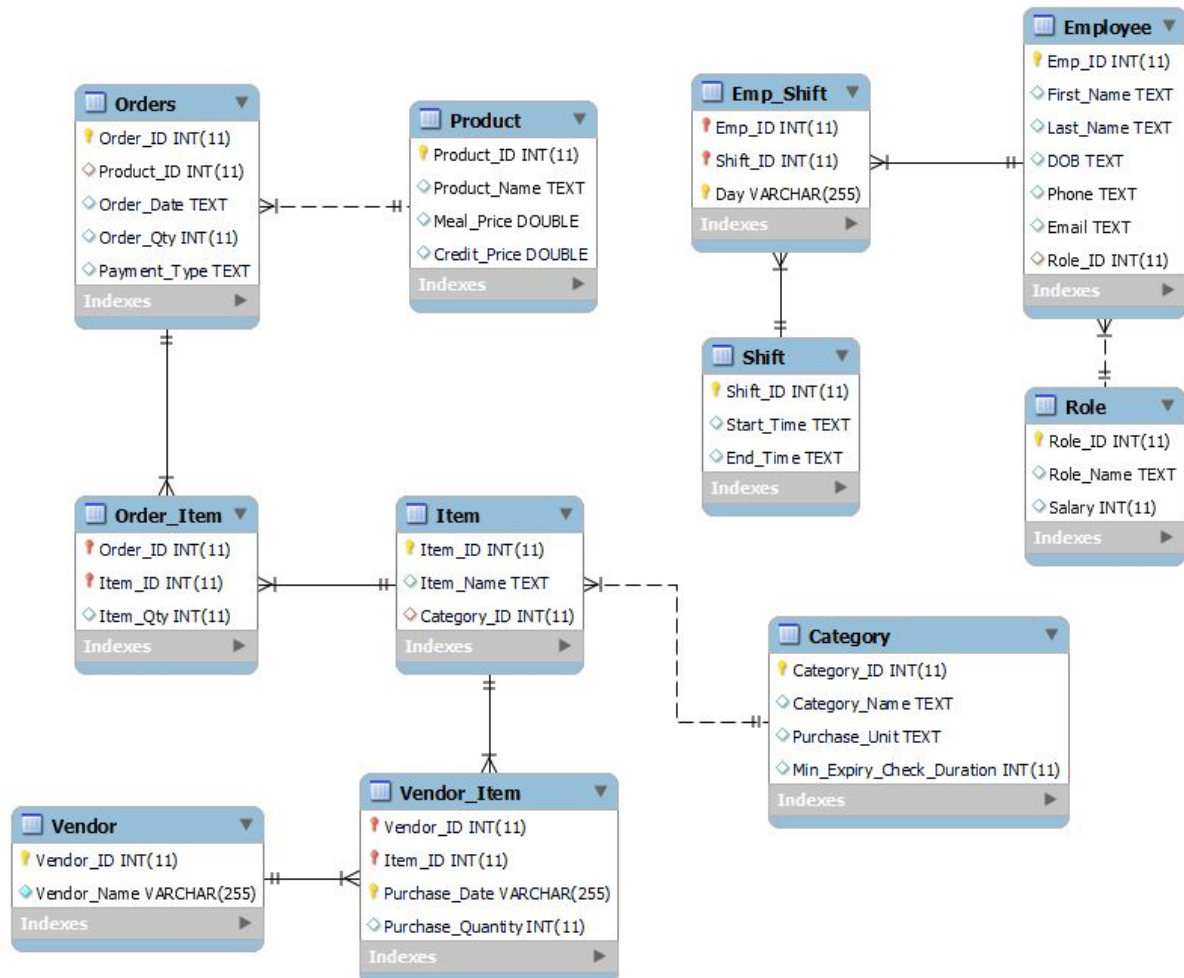
Hence, we created a new Vendor table to hold the Vendor related data and these tables were in 3NF due to no transitive dependencies.

Vendor_ID	Item_ID	Purchase_Date	Purchase_Quantity
8278	1083	10/23/2021	2
8278	1084	10/23/2021	3
8278	1085	10/23/2021	2
8278	1086	10/23/2021	5
8278	1021	10/23/2021	2
8278	1022	10/23/2021	2

Vendor ID	Vendor Name
6745	Noble Coffee and Tea Inc.
6009	Disposables MMDC
3112	Hanson Cold Storage
7634	Aunt Millies Bakery
5143	Piazza Cibus Fresh
1686	Piazza Indianapolis

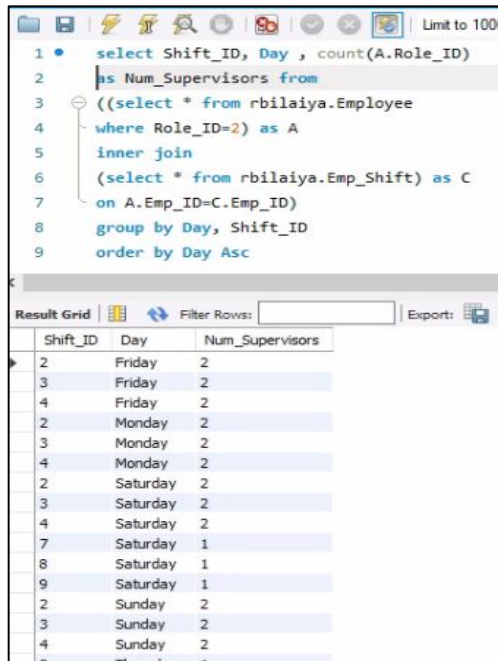
DATA WAREHOUSE:

Once data is normalized, we created the data warehouse in MYSQL to store the data and perform queries. Below is the ERD generated in MYSQL workbench:



BUSINESS QUESTIONS SOLVED USING SQL:

1. The current shift-wise distribution of employees in keeping with the daily demand? Does it require a change?



Query 1 SQL:

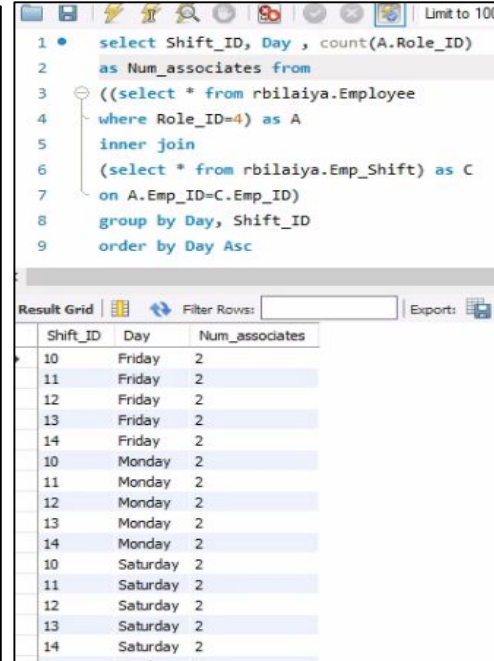
```

1 • select Shift_ID, Day , count(A.Role_ID)
2   as Num_Supervisors from
3   ((select * from rbilaiya.Employee
4    where Role_ID=2) as A
5   inner join
6    (select * from rbilaiya.Emp_Shift) as C
7   on A.Emp_ID=C.Emp_ID)
8  group by Day, Shift_ID
9  order by Day Asc
  
```

Result Grid:

Shift_ID	Day	Num_Supervisors
2	Friday	2
3	Friday	2
4	Friday	2
2	Monday	2
3	Monday	2
4	Monday	2
2	Saturday	2
3	Saturday	2
4	Saturday	2
7	Saturday	1
8	Saturday	1
9	Saturday	1
2	Sunday	2
3	Sunday	2
4	Sunday	2

Query 1



Query 2 SQL:

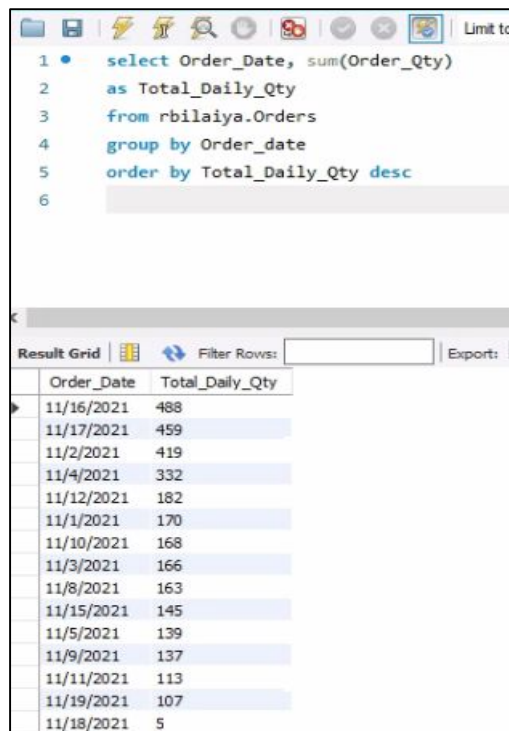
```

1 • select Shift_ID, Day , count(A.Role_ID)
2   as Num_associates from
3   ((select * from rbilaiya.Employee
4    where Role_ID=4) as A
5   inner join
6    (select * from rbilaiya.Emp_Shift) as C
7   on A.Emp_ID=C.Emp_ID)
8  group by Day, Shift_ID
9  order by Day Asc
  
```

Result Grid:

Shift_ID	Day	Num_associates
10	Friday	2
11	Friday	2
12	Friday	2
13	Friday	2
14	Friday	2
10	Monday	2
11	Monday	2
12	Monday	2
13	Monday	2
14	Monday	2
10	Saturday	2
11	Saturday	2
12	Saturday	2
13	Saturday	2
14	Saturday	2

Query 2



Query 3 SQL:

```

1 • select Order_Date, sum(Order_Qty)
2   as Total_Daily_Qty
3   from rbilaiya.Orders
4  group by Order_date
5  order by Total_Daily_Qty desc
6
  
```

Result Grid:

Order_Date	Total_Daily_Qty
11/16/2021	488
11/17/2021	459
11/2/2021	419
11/4/2021	332
11/12/2021	182
11/1/2021	170
11/10/2021	168
11/3/2021	166
11/8/2021	163
11/15/2021	145
11/5/2021	139
11/9/2021	137
11/11/2021	113
11/19/2021	107
11/18/2021	5

Query 3

Query 1: Query to find the number of supervisors who are working in each shift, every working day of the week.

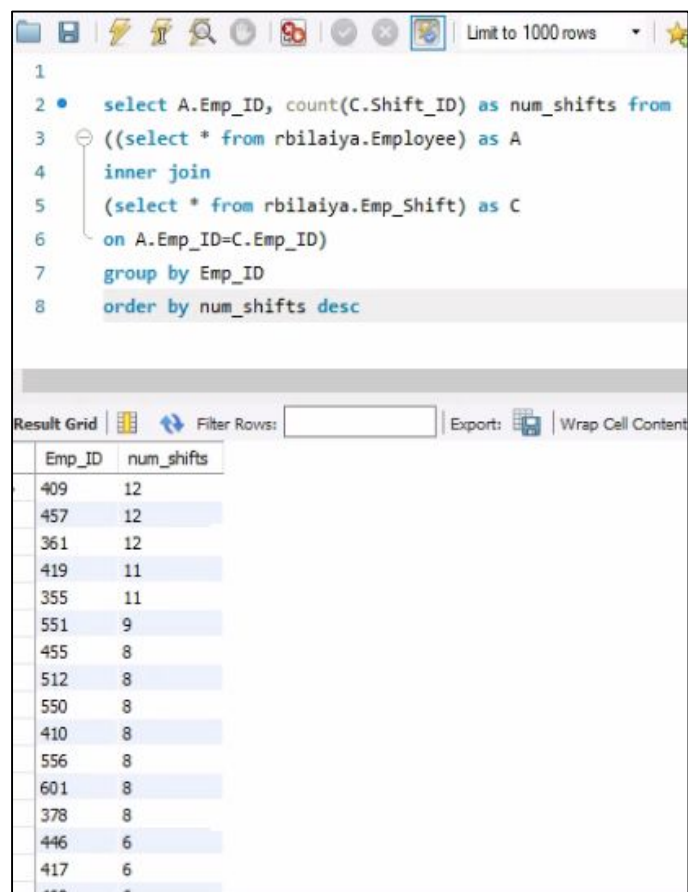
Query 2: Query to find the number of associates who are working in each shift, every working day of the week.

Query 3: Query to find the daily demand (total number of daily orders) at Lawson.

INSIGHTS AND RECOMMENDATION:

- Efficient and optimum distribution of employees with respect to the demand or rush in the Dining Hall is a pertinent problem
- For every shift, there are two supervisors and two associates consistently across all days.
- But daily demand is dynamic. It ranges from 488 to merely 5 on certain days
- Therefore, Lawson requires better distribution of resources

2. Are the employees being used to their full capacity?



```

1
2 • select A.Emp_ID, count(C.Shift_ID) as num_shifts from
3 ((select * from rbilaiya.Employee) as A
4 inner join
5 (select * from rbilaiya.Emp_Shift) as C
6 on A.Emp_ID=C.Emp_ID)
7 group by Emp_ID
8 order by num_shifts desc

```

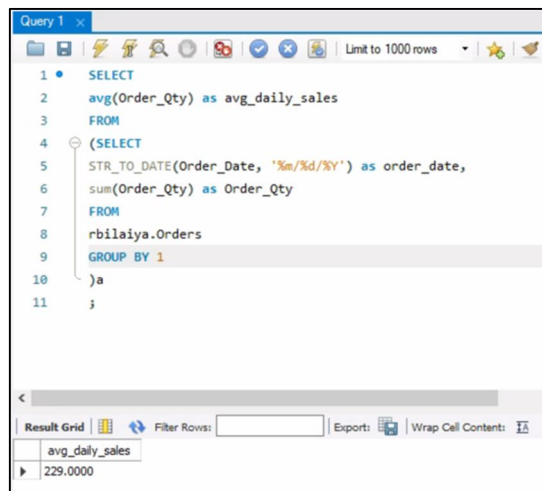
Emp_ID	num_shifts
409	12
457	12
361	12
419	11
355	11
551	9
455	8
512	8
550	8
410	8
556	8
601	8
378	8
446	6
417	6

Query: Query to find the number of shifts each an every employee working at Lawson is assigned to in a given working week.

INSIGHTS AND RECOMMENDATION:

- The distribution of shifts amongst the employees doesn't seem to be done in the most efficient manner
- There are certain employees covering as high as 12 shifts in working week whereas certain others are covering for just 1-2 shifts
- The business needs to make more equal distribution of workload and shifts to achieve better efficiency and reduce workload (i.e., put a capping to number of shifts)

3. What is the average daily sale in Lawson OTG?

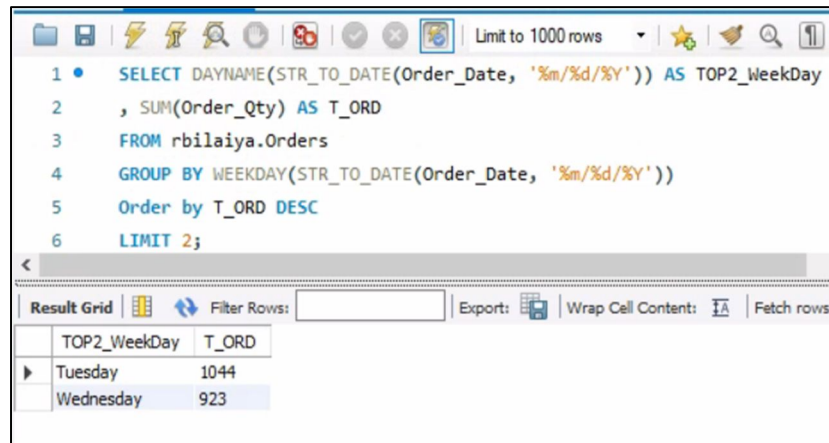


Query: We are finding the total sales at a date level and averaging the total across days to find the average daily sales

INSIGHTS AND RECOMMENDATION:

- There is a high variation in the order quantities across dates
- Even though the average daily sales is 229, it is not recommended to go with this number while planning inventory for a day because of the fluctuations
- The footfall, and hence order quantity depends on multiple other factors ranging from local festivals, games, exams and other factors.
- It is recommended to plan inventory for 3-4 days in advance

4. Which are the two days that are the busiest?



```

1 • SELECT DAYNAME(STR_TO_DATE(Order_Date, '%m/%d/%Y')) AS TOP2_WeekDay
2     , SUM(Order_Qty) AS T_ORD
3 FROM rbilaiya.Orders
4 GROUP BY WEEKDAY(STR_TO_DATE(Order_Date, '%m/%d/%Y'))
5 ORDER BY T_ORD DESC
6 LIMIT 2;

```

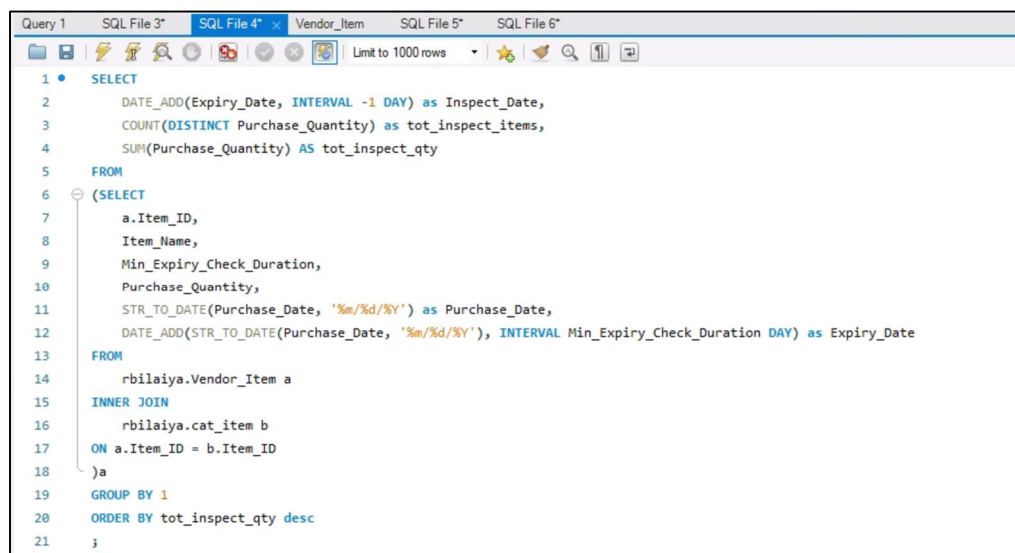
TOP2_WeekDay	T_ORD
Tuesday	1044
Wednesday	923

Query: We are finding the WeekDay corresponding to different order dates and counting the total sales for every working day (Monday-Friday). We can see Tuesday and Wednesday have highest sales compared to other days.

INSIGHTS AND RECOMMENDATION:

- We can see highest number of orders are placed on Tuesday and Wednesday compared to rest of the working days.
- As analysts we suggest having a greater number of backups and more employee to handle the workload on these 2 days.

5. How is the resource planning and management to account for food inspection and expiry?



```

1 • SELECT
2     DATE_ADD(Expiry_Date, INTERVAL -1 DAY) as Inspect_Date,
3     COUNT(DISTINCT Purchase_Quantity) as tot_inspect_items,
4     SUM(Purchase_Quantity) AS tot_inspect_qty
5 FROM
6     (SELECT
7         a.Item_ID,
8         Item_Name,
9         Min_Expiry_Check_Duration,
10        Purchase_Quantity,
11        STR_TO_DATE(Purchase_Date, '%m/%d/%Y') as Purchase_Date,
12        DATE_ADD(STR_TO_DATE(Purchase_Date, '%m/%d/%Y'), INTERVAL Min_Expiry_Check_Duration DAY) as Expiry_Date
13     FROM
14         rbilaiya.Vendor_Item a
15     INNER JOIN
16         rbilaiya.cat_item b
17     ON a.Item_ID = b.Item_ID
18     )a
19 GROUP BY 1
20 ORDER BY tot_inspect_qty desc
21 ;

```

Query: Inspection date for an item is the day before the expiry date. We are finding the inspection date for all the items and then aggregating at an inspection date level to find

the number of different items and total item quantity. The result is ordered by total item quantity in descending order

INSIGHTS AND RECOMMENDATION:

- Inspection date is usually the day before the stated expiration date of an item
- 29th October and 2nd November seem to be the busiest days when it comes to checking for item spoilage, hence may require a resource to clock longer hours than usual
- We see a total of 5 items that need to be checked for quality on 29th October which may entail new inventory purchase, hence inventory management should take this into consideration.

6. What are the food items that are most and least in demand?

Least in demand

Query 1 SQL File 3* SQL File 4* Vendor_Item SQL File 5*

```

1 SELECT
2   a.Product_ID,
3   Product_Name,
4   sum(Order_Qty) as demand
5 FROM
6   rbilaiya.Orders a, rbilaiya.Product b
7 WHERE a.Product_ID = b.Product_ID
8 GROUP BY 1,2
9 ORDER BY demand asc

```

Product_ID	Product_Name	demand
3	Water Cup	6

Most in demand

Query 1 SQL File 3* SQL File 4* Vendor_Item SQL File 5*

```

2   a.Product_ID,
3   Product_Name,
4   sum(Order_Qty) as demand
5 FROM
6   rbilaiya.Orders a, rbilaiya.Product b
7 WHERE a.Product_ID = b.Product_ID
8 GROUP BY 1,2
9 ORDER BY demand desc
10 LIMIT 1

```

Product_ID	Product_Name	demand
17	OTG Lunch Box	623

Query: We are finding the products that are being bought the most based on order quantity(highest) and bought the least based on order quantity (lowest)

INSIGHTS AND RECOMMENDATION

- It is clear that OTG lunch box and Bloomington Breakfast are the most in-demand products. The former is there because of the short time it requires to get it ready while the latter is present because of the customizability.
- Water is not demanded much which suggests that we need manage the inventory of the same frequently and since it does not go bad in a short time, it could be purchased at bulk

7. What are the recommended bestselling combinations?

```

1 • DROP TABLE IF EXISTS Product_Item_Qty_Temp;
2 • CREATE TEMPORARY TABLE Product_Item_Qty_Temp AS
3   SELECT ord.Product_id, Item.Category_ID, Item.Item_id, SUM(PI.Item_Qty) AS famous_item
4   FROM Orders ord
5   INNER JOIN ProdItem PI ON ord.order_id=PI.order_id
6   LEFT JOIN Item ON PI.Item_ID=Item.Item_ID
7   GROUP BY ord.Product_id , Item.Category_ID, Item.Item_id
8   ORDER BY ord.Product_id , Item.Category_ID, Item.Item_id, famous_item;
9
10 • DROP TABLE IF EXISTS Product_Fms_Item_Temp;
11 • CREATE TEMPORARY TABLE Product_Fms_Item_Temp AS
12   SELECT famous.Product_id, famous.Category_ID, max(famous_item) fms
13   FROM Product_Item_Qty_Temp famous
14   GROUP BY famous.Product_id, famous.Category_ID;

8 • SELECT P.Product_Name, Item.Item_Name
9   FROM Product_Item_Qty_Temp T
10  INNER JOIN Product_Fms_Item_Temp T2 ON T.Product_id=T2.Product_id
11  AND T.Category_ID=T2.Category_ID AND T.famous_item=T2.fms
12  INNER JOIN Item ON Item.Item_id=T.Item_ID
13  INNER JOIN Product P ON P.Product_ID=T.Product_ID
14  ORDER BY P.Product_Name;

```

Query: We created 2 temp tables; 1st table gives the total number of item quantity for every product on a category basis. Out of this we got the item in every category (bread, meat and cheese) which is being sold most for every product.

Product_Name	Item_Name
▶ Blooming Breakfast	Bacon Pre Cooked
Blooming Breakfast	Everything Bagel
Blooming Breakfast	Cheese Provolone Sliced
California Dream	Cheese Pepper Jack
California Dream	Cheese Provolone Sliced
California Dream	Roll Ciabetta White Sliced
California Dream	Sandwich Turkey Cheddar
Lawson Club	Cheese Cheddar Mild
Lawson Club	Roll Ciabetta White Sliced
Veg n wrap	Cheese Provolone Sliced
Veg n wrap	Wheat Bread

INSIGHTS AND RECOMMENDATION:

- According to our analysis, we recommend Lawson to high stock of provolone cheese as it is the bestselling item for 3 products.
- In case of breads, white bread and everything bagels are the most used items.

8. What is the most frequent payment mode?

The screenshot shows a SQL query in a text editor and its corresponding result grid. The query is: `SELECT Payment_Type as Most_Frequent_Payment, COUNT(Payment_Type) from Orders group by Most_Frequent_Payment order by COUNT(Payment_Type) desc limit 1;` The result grid has two columns: 'Most_Frequent_Payment' and 'COUNT(Payment_Type)'. The first row shows 'Meal/Credit' with a count of 152.

Most_Frequent_Payment	COUNT(Payment_Type)
Meal/Credit	152

Query- Most_Frequent_Payment speaks for itself as it denotes the payment method that has been used the most. We try to check our Orders table and retrieve the total number of times each payment method has been used.

INSIGHTS AND RECOMMENDATION:

- It is apparent that the most popular payment method is the Meal/Credit.
- This essentially means that a particular food item has been sold via Meal swipes as well as cards.
- While some items are limited to just meal swipes and others to just cards, it will draw in more of a buyer base if items are made available under both the categories.

9. Which vendors supply the categories of items that are purchased the most ordered by the quantity they supply and frequency they supply?

The screenshot shows a complex SQL query involving multiple joins and subqueries. The query is: `SELECT C.Category_Name, V.Vendor_Name, MAX(total_sum) AS MAX_Vendor_Qty FROM (SELECT Vendor_Item.Vendor_ID, SUM(Vendor_Item.Purchase_Quantity) AS total_sum, Category.Category_ID FROM Vendor_Item INNER JOIN Item ON Vendor_Item.Item_ID = Item.Item_ID INNER JOIN Category ON Item.Category_ID = Category.Category_ID GROUP BY Vendor_ID, Category_ID order by SUM(Purchase_Quantity) DESC) AS TEMP INNER JOIN Category C ON TEMP.Category_ID = C.Category_ID INNER JOIN Vendor V ON TEMP.Vendor_ID = V.Vendor_ID group by TEMP.Category_ID;` The result grid has three columns: 'Category_Name', 'Vendor_Name', and 'MAX_Vendor_Qty'. The data is as follows:

Category_Name	Vendor_Name	MAX_Vendor_Qty
Beverages	Dean Foods/Schenck's Dairy	41
Eatery Set/Containers	Disposables MMDC	950
Meat	Instantwhip Foods	5
Toppings, Dips, Sauces, Dressings	Piazza Indianapolis	111
Pre Cooked Non-Veg	U.S. Foodservice	450
Confectionary	Disposables MMDC	150
Breads	Aunt Millies Bakery	340

Query- We create a temporary table that is a join of the Vendor_Item, Category and Item table based on the Item_ID field. Now we order this by the total sum of Purchase quantity and group it by Vendor_ID and Category_ID. On top of this temp table we again do a join of the Category table and Vendor table based on Category ID and Vendor ID and group it again by Category ID to obtain the Vendor who has sold the maximum quantity under each category.

INSIGHTS AND RECOMMENDATION:

- As a result, we get a list of the categories where a particular vendor supplies the maximum quantity.
- This will help us choose the best vendor to make a purchase from the next time a product under a particular category has to be purchased since its clearly ordered in a large quantity in comparison.

10. Which item and category should be replenished soon based on the difference between order quantity and supply quantity?

Query 1 SQL File 3* SQL File 4* Vendor_Item SQL File 5* SQL File 6* x

```

1  SELECT
2      item_ID,
3      Item_Name,
4      stock_qty - used_qty as rem_qty
5  FROM
6      (SELECT
7          a.*,
8          b.Purchase_Quantity as stock_qty,
9          c.Item_Qty as used_qty
10     FROM
11         Item a
12     INNER JOIN
13         Vendor_Item b
14     ON a.Item_ID = b.Item_ID
15     INNER JOIN
16         Order_Item c
17     ON a.Item_ID = c.Item_ID
18     WHERE stock_qty > used_qty)
19     a
20 GROUP BY 1,2,3
21 ORDER BY rem_qty asc

```

item_ID	Item_Name	rem_qty
1008	Cheese Pepper Jack	0
1006	Cheese Cheddar Mild	0
1009	Cheese Provolone Sliced	0
1042	Roll Ciabetta White Sliced	0
1003	Everything Bagel	0
1006	Cheese Cheddar Mild	1
1008	Cheese Pepper Jack	1
1003	Everything Bagel	1
1008	Cheese Pepper Jack	2
1009	Cheese Provolone Sliced	2
1002	Plain Bagel	2
1006	Cheese Cheddar Mild	2
1003	Everything Bagel	2
1006	Cheese Cheddar Mild	3
1042	Roll Ciabetta White Sliced	3
1009	Cheese Provolone Sliced	3
1003	Everything Bagel	3
1008	Cheese Pepper Jack	3
1006	Cheese Cheddar Mild	4
1009	Cheese Provolone Sliced	4
1008	Cheese Pepper Jack	4
1002	Plain Bagel	4
1042	Roll Ciabetta White Sliced	5
1009	Cheese Provolone Sliced	5

Query: We are finding the items that need to be replenished soon by finding the difference between purchase quantity and order quantity for each item

INSIGHTS AND RECOMMENDATION

- It is crucial to check for out of stock quantities in order to prevent items from being excluded from the menu, and hence maximize the revenue.
- From the snippet, it is clear that cheese and bagel are running out of stock which puts the desserts part of the menu at risk.
- Also, this could be automated as it is important to check for quantities of items that might run out of stock soon

RECOMMENDATION:

- Employees to be put on shifts based on the level of food order demand in the week, instead of uniform distribution.
- Daily inventory should be checked to take care of days where multiple items are required in advance. A recurring reminder in calendar to order would help.
- All the menu options should be provided on both meal swipe and credit card options as both are very widely and extensively used while ordering.
- Having a list of vendors for specific food items in bulk should be provided to the management staff including student manager for crisis management.

IMPROVEMENTS:

- ✓ The tally of inventory utilization is updated at a daily level after taking stock of the inventory in the end. This could be automated by doing it at a transaction level.
- ✓ A link between employee and orders table is not present. This could be added for more effective human resource management.
- ✓ Stored procedures could be implemented to automatically calculate the expiry duration of items at a daily level.
- ✓ Beverages are very customizable and the same can be implemented in our database as it is designed to scale.