RESTAURANT MANAGEMENT SYSTEM

Prepared for Eng Rana Salah

Team Members:

MOHAMMAD EMAD
MENNA AM R
ROFAIDA BELAL
OMAR AHMED
ABDALLAH IBRAHIM

1 CONTENTS

2 INTRODUCTION:

It’s become well known that in today’s world, data is one of the most important things in our day-to-day lives. From data alone, we can learn a human’s entire life and predict their future. However, such data comes in a raw form that doesn’t help provide any insights, and that’s where we come in. We have to clean the data and organize it in order to come up with an understandable insight that could help further improve the business, and the business we’re doing the data for is a restaurant.

Here we’re tasked with a plan to design and implement a dimensional model for storing menu items, customer orders, inventory, employee information, and sales data. Menu Items Dimension stores details about the menu items offered by the restaurant. Inventory Dimension stores details about inventory items such as item ID, name, quantity in stock, supplier information, etc. Employee Dimension contains information about employees such as employee ID, name, work duration, etc. Sales Fact Table captures transactional data related to sales. It typically includes foreign keys referencing the dimension tables, as well as measures such as quantity sold, total sales amount, discounts applied, timestamps, etc. This table serves as the central hub for analyzing sales data. Order Dimension Depending on the complexity of the system, you might create a separate dimension table for orders. This table could contain attributes like order ID, order date, payment method, etc. Date Dimension, If time-based analysis is important, a time dimension table can be included with attributes such as date, day of the week, month, quarter, year, etc.

Establish relationships between the fact table and dimension tables using foreign keys. For example, the sales fact table would have foreign keys referencing the menu items, employees involved in the sale, etc. Define the level of granularity for the fact table. In this case, it could be at the level of individual sales transactions, capturing details of each sale. Consider creating aggregated tables for faster querying and reporting. Aggregations can be based on different dimensions and measures, such as total sales by item, total sales by customer, etc.

To implement the features we need in this project, we can use toad to write PL/SQL code for inventory management, to track stock levels and ingredient usage. ***-The way we implemented it-***

We implemented SQL queries designed to analyze monthly sales data to gain insights into sales performance, including sales velocity, transaction count, and sales growth. We implemented another query that provides a breakdown of sales by order type for each month, aiding in understanding the distribution of sales across different types of orders over time. Another one that helps in identifying peak hours for morning and night shifts each month, enabling effective scheduling and resource management. A query that identifies items that have been purchased only once. It calculates the total quantity of each item and filters items with a total quantity of 1. The result includes the item name, category, price, and the number of stocks. It's sorted by category. A query that calculates the growth of selling products over time. It aggregates the total sales quantity for each month and calculates the previous year's sales amount. The difference between the current year's sales and the previous year's sales gives the sales growth for each month. A query that helps our business identify products that perform poorly each month, enabling us to adjust inventory, marketing strategies, or even consider discontinuing such products to optimize business performance. A query that presents the sales of each category. It calculates the total quantity, total price, and sales revenue for each category. Additionally, it computes the percentage rank of sales revenue among all categories. A query that identifies the most selling meal in each category. It ranks meals based on their total sales within each category and selects the meal with the highest sales in each category. A query that provides the most selling meal in each category along with its revenue. It calculates the total revenue for each meal by subtracting the total ingredient cost from its total sales. The results are grouped by meal name, category, total sales, total ingredient cost, and revenue.

3 DATA SOURCES:

We've gotten our data from a big fish restaurant chain that provides hundreds of seafood menu items for customers. The data we've acquired include the following:

-Employee table: where it includes the name of the employees, their ID, their shifts, phone numbers, job title, age, and performance evaluation.

-Attendance table: where it includes the employee ID and their attendance on each day.

-Inventory table: where it includes inventory ID, item name, code, item group, unit, stock placement, supplier and their name.

-Order header table: where it includes order ID, order date and time, order type, and the total money.

-Reservation table: where it includes reservation ID, the name of the person that reserved, the time, the table they reserved, the occasion, and the size of the party.

-Menu table: where it contains menu item names, their ID, their category, their availability, and their descriptions.

-Item sales table: where it includes the item name, their category, their quantity, their price, their total, and the amount sold a month.

-Ingredient table: where it includes the inventoryID, meal ID, item name, item ID, price, category, item name, code, unit, ingredient ID, ingredient category, stock, cost, and quantity.

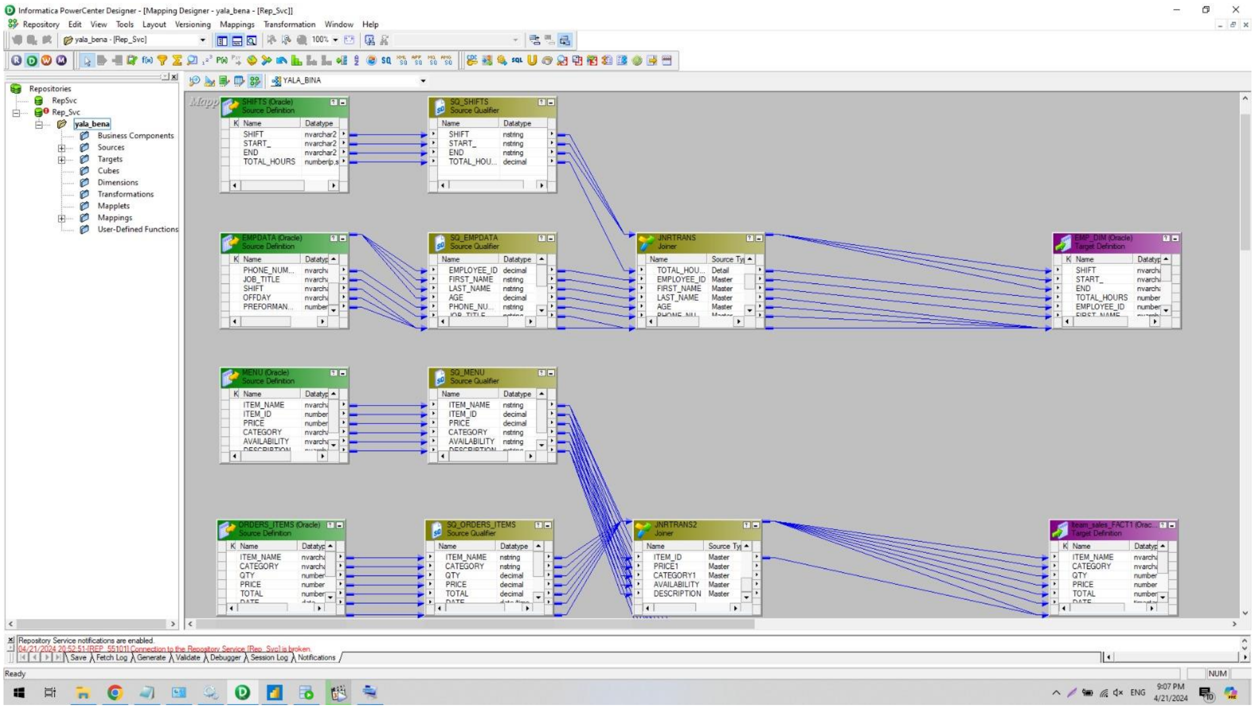
We chose this data as it fits to our needs for us to create them a management system for their restaurant, as well as a dashboard to learn their trends and to be able to take business decisions to help further improve their business.

4 DATA TRANSFORMATION:

First step we did was clean our data using excel power query as any data needs cleaning in order to be able to work efficiently on it. First we translated the data from Arabic to English, then we trimmed data that we have no use for, removed duplicates, corrected errors, standardized data types. Then we started formatting columns if it's integer, alphanumerical, date or currency. We used unpivoting function to further show clearer data in order to be able to use, as well as merged tables together and split other tables. We used data aggregations to get totals, averages, substractions and so on. We filtered and sorted data in order to get specific criterias to help make the data clearer. Used functions such as if, vlookup on excel to manipulate the data to make the visualization easier. We created data visualization on power BI, creating charts and graphs to visualize patterns and understand the data more clearly to make better decisions going down the line. Also generated reports and dashboards for easier analysis, such as statistical analysis to see which menu item for example is least sold and if it's worth keeping or not. We also used informatica to join tables for our model.

5 DATA WAREHOUSE DATA MODEL:

We used galaxy schema for our data model, which is a hybrid between the star schema and the snowflake schema. In a Galaxy schema, you have multiple star schemas that are interconnected, resembling a galaxy with multiple stars. The Snowflake schema is a type of star schema, but with normalized dimensions. It's called "snowflake" because of its shape, where the fact table is at the center surrounded by multiple dimension tables, and each dimension can be further broken down into sub-dimensions, resembling a snowflake. The star schema is a type of data warehouse schema where data is organized into fact tables surrounded by dimension tables. It's called a "star" schema because the diagram representing it resembles a star, with one central fact table connected to multiple dimension tables radiating outwards.



Informatica PowerCenter Designer - [Mapping Designer - yala_bena - [Rep_Svc]]

SHIFTS (Oracle)
Source Definition

K	Name	Datatype
	SHIFT	nvarchar2
	START_	nvarchar2
	END	nvarchar2
	TOTAL_HOURS	number(p,s

SQ_SHIFTS
Source Qualifier

Name	Datatype
SHIFT	nstring
START_	nstring
END	nstring
TOTAL_HOU...	decimal

EMPDATA (Oracle)
Source Definition

K	Name	Datatyp
	PHONE_NUM...	nvarch
	JOB_TITLE	nvarch
	SHIFT	nvarch
	OFFDAY	nvarch
	PREFORMAN...	number

SQ_EMPDATA
Source Qualifier

Name	Datatype
EMPLOYEE_ID	decimal
FIRST_NAME	nstring
LAST_NAME	nstring
AGE	decimal
PHONE_NU...	nstring

JNRTRANS
Joiner

Name	Source Ty
TOTAL_HOU...	Detail
EMPLOYEE_ID	Master
FIRST_NAME	Master
LAST_NAME	Master
AGE	Master

EMP_DIM (Oracle)
Target Definition

K	Name	Datatyp
	SHIFT	nvarch
	START_	nvarch
	END	nvarch
	TOTAL_HOURS	number
	EMPLOYEE_ID	number

MENU (Oracle)
Source Definition

K	Name	Datatyp
	ITEM_NAME	nvarch
	ITEM_ID	number
	PRICE	number
	CATEGORY	nvarch
	AVAILABILITY	nvarch

SQ_MENU
Source Qualifier

Name	Datatype
ITEM_NAME	nstring
ITEM_ID	decimal
PRICE	decimal
CATEGORY	nstring
AVAILABILITY	nstring

ORDERS_ITEMS (Oracle)
Source Definition

K	Name	Datatyp
	ITEM_NAME	nvarch
	CATEGORY	nvarch
	QTY	number
	PRICE	number
	TOTAL	number

SQ_ORDERS_ITEMS
Source Qualifier

Name	Datatype
ITEM_NAME	nstring
CATEGORY	nstring
QTY	decimal
PRICE	decimal
TOTAL	decimal

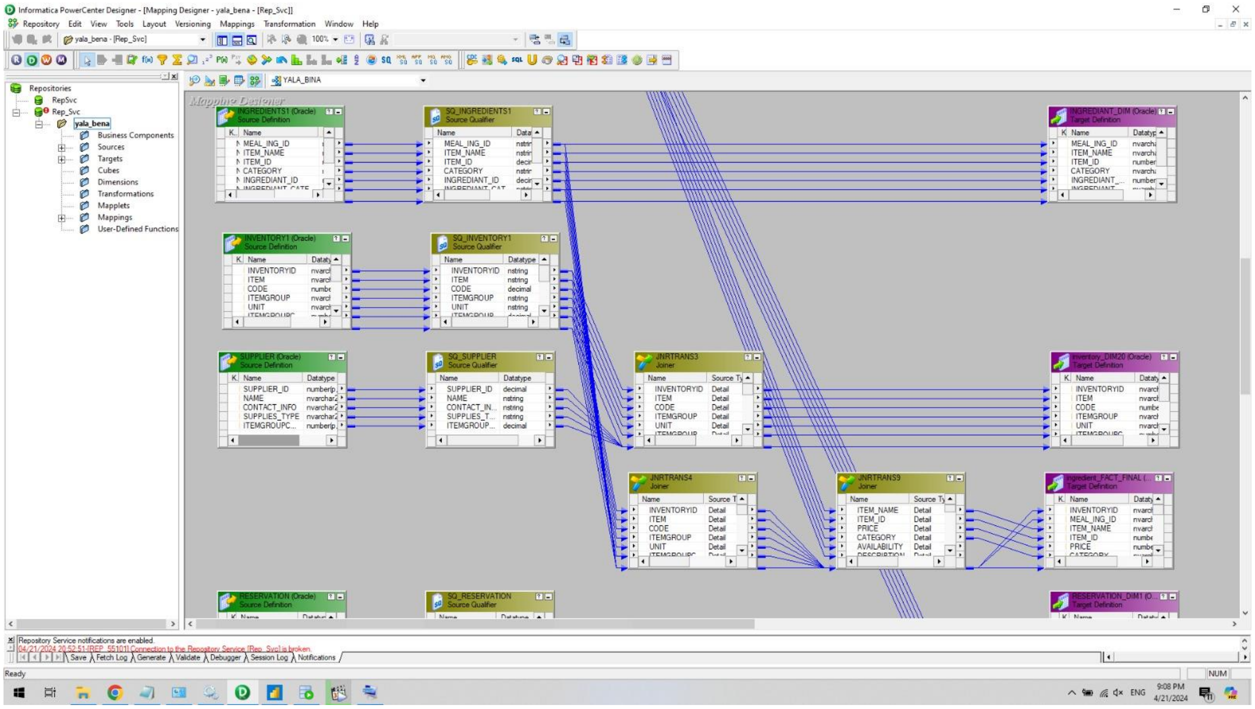
JNRTRANS2
Joiner

Name	Source Ty
ITEM_ID	Master
PRICE1	Master
CATEGORY1	Master
AVAILABILITY	Master
DESCRIPTION	Master

team_sales_FACT1 (Orac...
Target Definition

K	Name	Datatyp
	ITEM_NAME	nvarch
	CATEGORY	nvarch
	QTY	number
	PRICE	number
	TOTAL	number

```
Repository Service notifications are enabled.
04/21/2024 20:52:51-[REP_55101] Connection to the Repository Service [Rep_Svc] is broken.
```

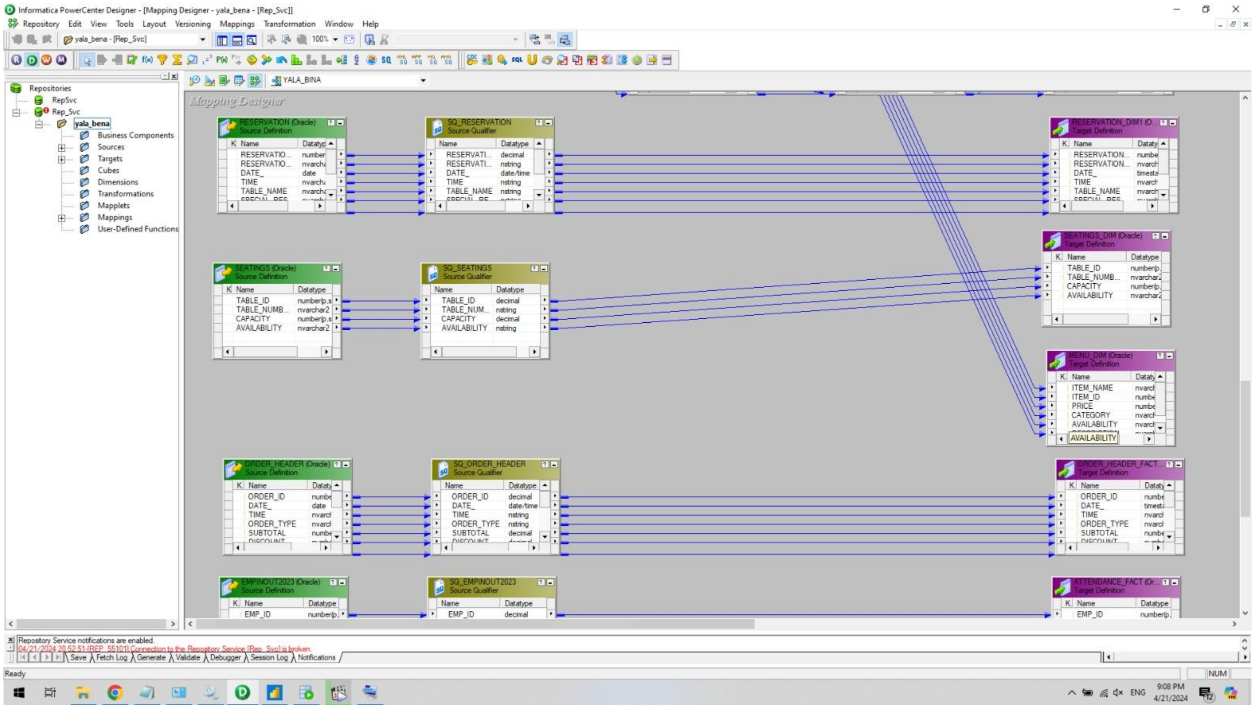


Informatica PowerCenter Designer - [Mapping Designer - yala_bena - [Rep_Svc]]

Repository Service notifications are enabled.
04/21/2024 20:52:51-[REP_55101] Connection to the Repository Service [Rep_Svc] is broken.

Ready

Repository Edit View Tools Layout Versioning Mappings Transformation Window Help
yala_bena - [Rep_Svc]
100%
YALA_BINA
Repositories
RepSvc
Rep_Svc
yala_bena
Business Components
Sources
Targets
Cubes
Dimensions
Transformations
Mapplets
Mappings
User-Defined Functions
Mapping Designer
INGREDIENTS1 (Oracle) Source Definition
SQ_INGREDIENTS1 Source Qualifier
INGREDIANT_DIM (Oracle) Target Definition
MEAL_ING_ID
ITEM_NAME
ITEM_ID
CATEGORY
INGREDIANT_ID
INVENTORY1 (Oracle) Source Definition
SQ_INVENTORY1 Source Qualifier
INVENTORYID
ITEM
CODE
ITEMGROUP
UNIT
SUPPLIER (Oracle) Source Definition
SQ_SUPPLIER Source Qualifier
SUPPLIER_ID
NAME
CONTACT_INFO
SUPPLIES_TYPE
ITEMGROUPC...
JNRTRANS3 Joiner
Inventory_DIM20 (Oracle) Target Definition
JNRTRANS4 Joiner
JNRTRANS9 Joiner
ITEM_NAME
ITEM_ID
PRICE
CATEGORY
AVAILABILITY
Ingredient_FACT_FINAL (... Target Definition
INVENTORYID
MEAL_ING_ID
ITEM_NAME
ITEM_ID
PRICE
RESERVATION (Oracle) Source Definition
SQ_RESERVATION Source Qualifier
RESERVATION_DIM1 (O... Target Definition
Save Fetch Log Generate Validate Debugger Session Log Notifications
NUM
9:08 PM 4/21/2024
ENG

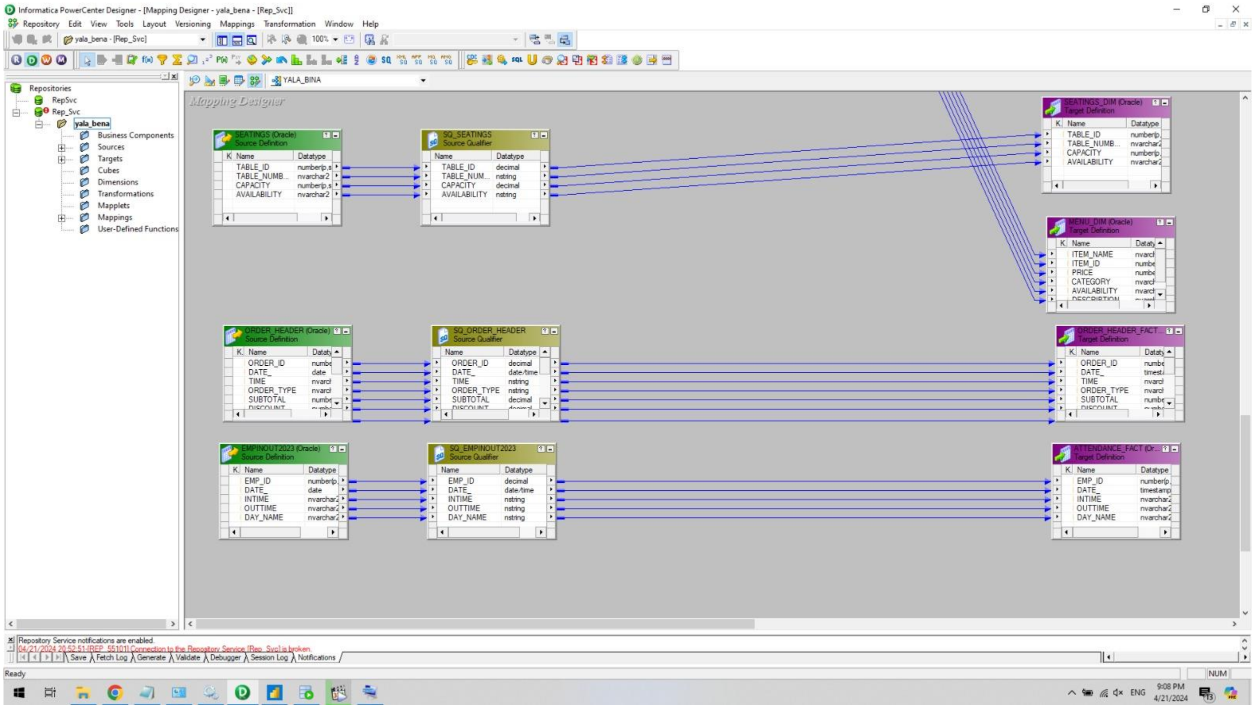


Informatica PowerCenter Designer - [Mapping Designer - yala_bena - [Rep_Svc]]

Repository Service notifications are enabled.
04/21/2024 20:52:51-[REP_55101] Connection to the Repository Service [Rep_Svc] is broken.

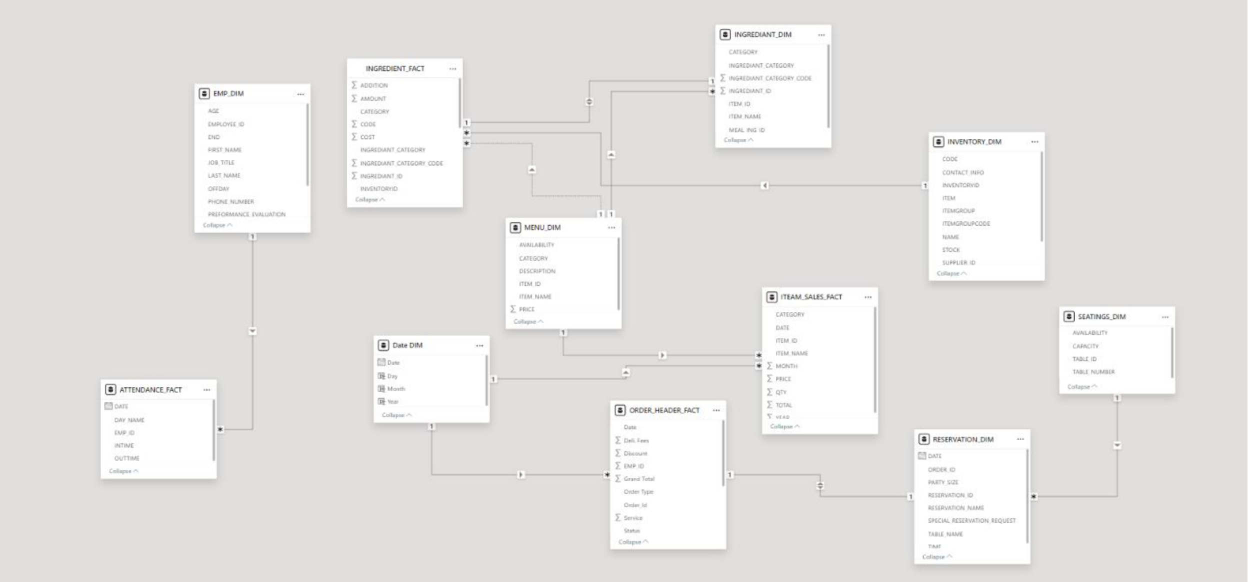
Ready

Repository Edit View Tools Layout Versioning Mappings Transformation Window Help
yala_bena - [Rep_Svc]
100%
YALA_BINA
Mapping Designer
RESERVATION (Oracle) Source Definition
SQ_RESERVATION Source Qualifier
RESERVATION_DIM1 (O... Target Definition
RESERVATIO...
DATE_
TIME
TABLE_NAME
SEATINGS (Oracle) Source Definition
SQ_SEATINGS Source Qualifier
SEATINGS_DIM (Oracle) Target Definition
TABLE_ID
TABLE_NUMB...
CAPACITY
AVAILABILITY
MENU_DIM (Oracle) Target Definition
ITEM_NAME
ITEM_ID
PRICE
CATEGORY
AVAILABILITY
ORDER_HEADER (Oracle) Source Definition
SQ_ORDER_HEADER Source Qualifier
ORDER_HEADER_FACT... Target Definition
ORDER_ID
DATE_
TIME
ORDER_TYPE
SUBTOTAL
EMPINOUT2023 (Oracle) Source Definition
SQ_EMPINOUT2023 Source Qualifier
ATTENDANCE_FACT (Or... Target Definition
EMP_ID
Save Fetch Log Generate Validate Debugger Session Log Notifications
NUM
9:08 PM 4/21/2024
ENG


Informatica PowerCenter Designer - [Mapping Designer - yala_bena - [Rep_Svc]]
Repository Edit View Tools Layout Versioning Mappings Transformation Window Help
Repositories
RepSvc
Rep_Svc
yala_bena
Business Components
Sources
Targets
Cubes
Dimensions
Transformations
Mapplets
Mappings
User-Defined Functions
YALA_BINA
Mapping Designer
SEATINGS (Oracle) Source Definition
SQ_SEATINGS Source Qualifier
SEATINGS_DIM (Oracle) Target Definition
MENU_DIM (Oracle) Target Definition
ORDER_HEADER (Oracle) Source Definition
SQ_ORDER_HEADER Source Qualifier
ORDER_HEADER_FACT Target Definition
EMPINOUT2023 (Oracle) Source Definition
SQ_EMPINOUT2023 Source Qualifier
ATTENDANCE_FACT (Or Target Definition
TABLE_ID
TABLE_NUMB...
CAPACITY
AVAILABILITY
ITEM_NAME
ITEM_ID
PRICE
CATEGORY
DESCRIPTION
ORDER_ID
DATE_
TIME
ORDER_TYPE
SUBTOTAL
DISCOUNT
EMP_ID
INTIME
OUTTIME
DAY_NAME
Save Fetch Log Generate Validate Debugger Session Log Notifications
Ready
NUM
9:08 PM
4/21/2024

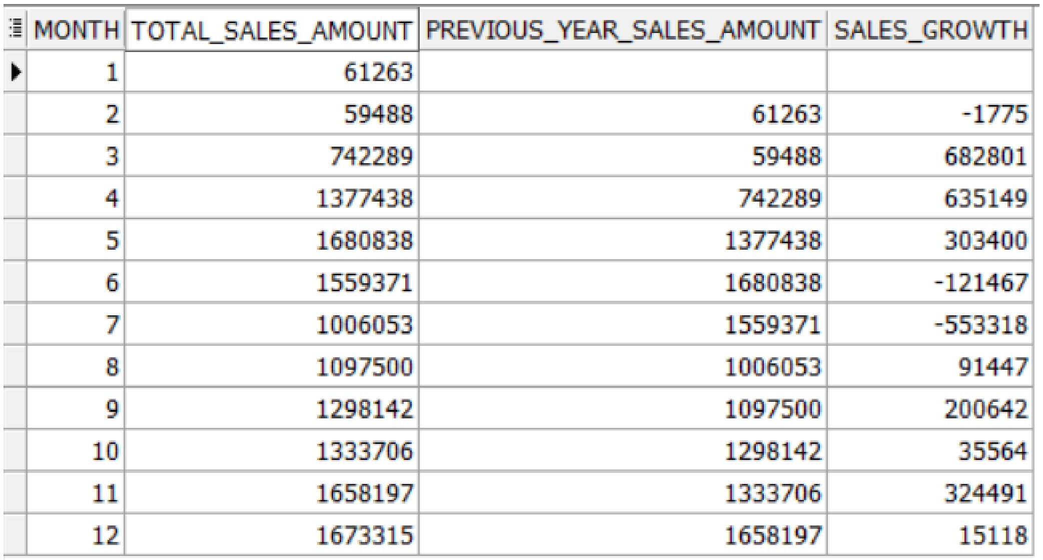
Repository Service notifications are enabled.
04/21/2024 20:52:51 [REP_55101] Connection to the Repository Service [Rep_Svc] is broken.

We have 4 fact tables which are (Ingredient, Order header, Item sales, Attendance.) and 7 dimensions (Date, Inventory, Seatings, Reservation, Menu, Ingredient, Employees.)



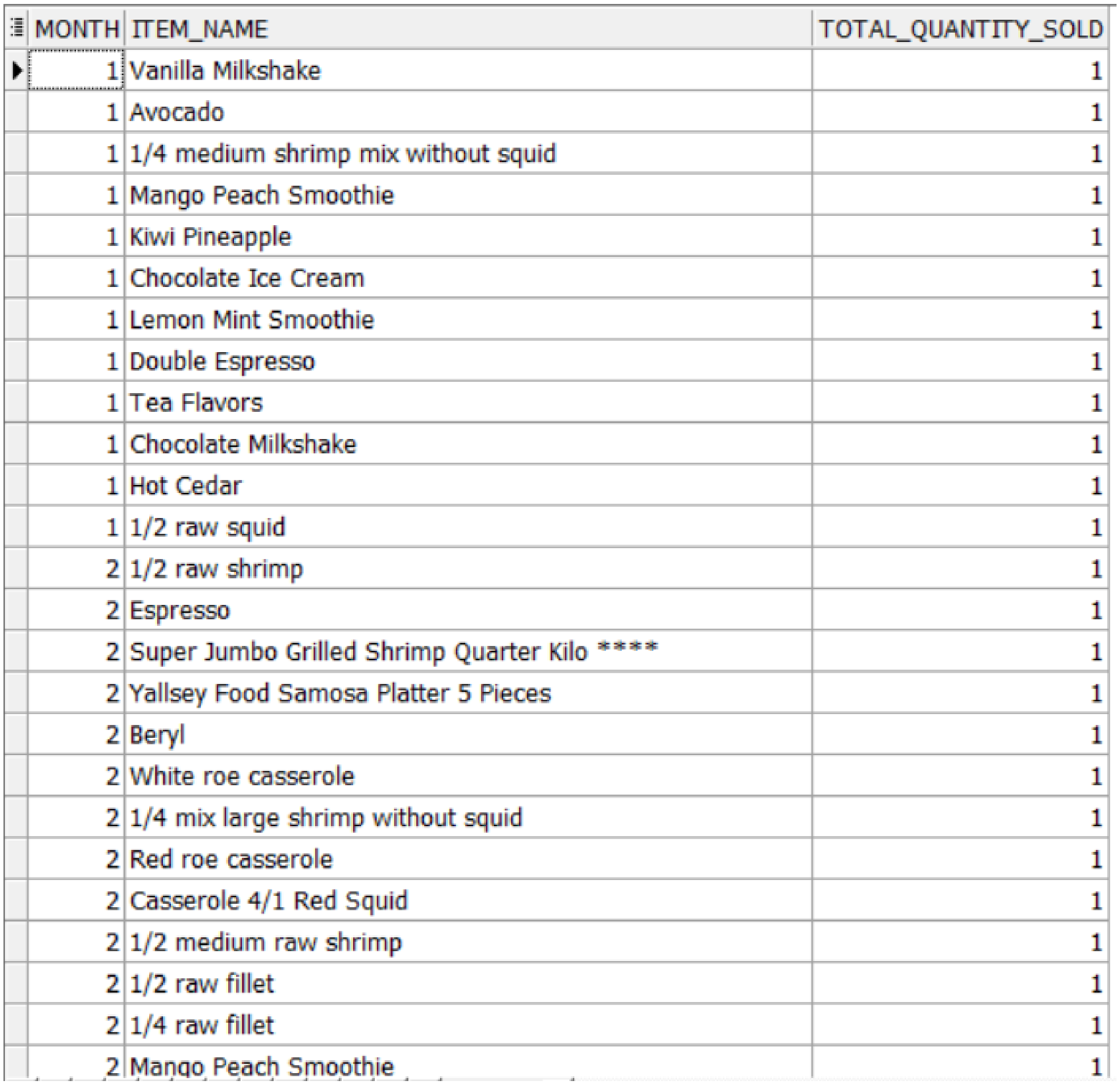
6 BI QUERIES:

```
        ----- selling products growth
SELECT
  month,
  SUM(QTY) AS Total_Sales_Amount,
  LAG(SUM(QTY)) OVER (ORDER BY month) AS Previous_Year_Sales_Amount,
    (SUM(QTY) - LAG(SUM(QTY)) OVER (ORDER BY month)) AS Sales_Growth
FROM data.order_Sales
GROUP BY month;
```



MONTH	TOTAL_SALES_AMOUNT	PREVIOUS_YEAR_SALES_AMOUNT	SALES_GROWTH
1	61263		
2	59488	61263	-1775
3	742289	59488	682801
4	1377438	742289	635149
5	1680838	1377438	303400
6	1559371	1680838	-121467
7	1006053	1559371	-553318
8	1097500	1006053	91447
9	1298142	1097500	200642
10	1333706	1298142	35564
11	1658197	1333706	324491
12	1673315	1658197	15118

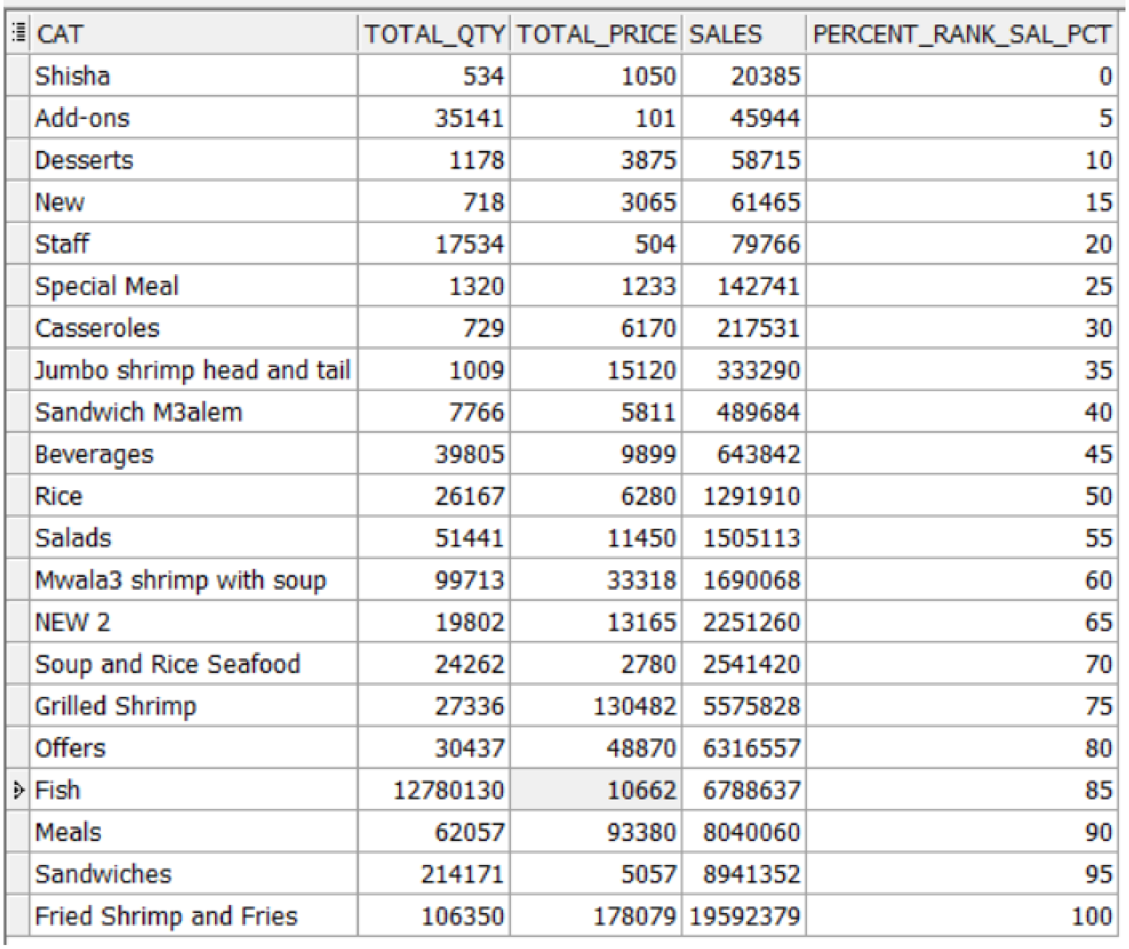
```
--what is the least selling product in each month? -- business can avoid
WITH MonthlyProductSales AS (
    SELECT

        Month,
        Item_Name,
        SUM(QTY) AS Total_Quantity_Sold
    FROM order_sales
    GROUP BY  Month, Item_Name
),
LeastSellingMonth AS (
    SELECT

        Month,
        MIN(Total_Quantity_Sold) AS Min_Quantity_Sold
    FROM MonthlyProductSales
    GROUP BY  Month
),
LeastSellingProduct AS (
    SELECT
        Month,
        Item_Name,
        Total_Quantity_Sold
    FROM MonthlyProductSales
    WHERE (Month, Total_Quantity_Sold) IN (
        SELECT Month, Min_Quantity_Sold
        FROM LeastSellingMonth
    )
)
SELECT
    Month,
    Item_Name,
    Total_Quantity_Sold
FROM LeastSellingProduct
ORDER BY Total_Quantity_Sold, Month;
```



MONTH	ITEM_NAME	TOTAL_QUANTITY_SOLD
1	Vanilla Milkshake	1
1	Avocado	1
1	1/4 medium shrimp mix without squid	1
1	Mango Peach Smoothie	1
1	Kiwi Pineapple	1
1	Chocolate Ice Cream	1
1	Lemon Mint Smoothie	1
1	Double Espresso	1
1	Tea Flavors	1
1	Chocolate Milkshake	1
1	Hot Cedar	1
1	1/2 raw squid	1
2	1/2 raw shrimp	1
2	Espresso	1
2	Super Jumbo Grilled Shrimp Quarter Kilo ****	1
2	Yallsey Food Samosa Platter 5 Pieces	1
2	Beryl	1
2	White roe casserole	1
2	1/4 mix large shrimp without squid	1
2	Red roe casserole	1
2	Casserole 4/1 Red Squid	1
2	1/2 medium raw shrimp	1
2	1/2 raw fillet	1
2	1/4 raw fillet	1
2	Mango Peach Smoothie	1

```
--lets take alook on the sales of each category
SELECT
    CAT,
     Total_QTY, Total_price, Sales,
    ROUND(PERCENT_RANK() OVER ( ORDER BY Sales) * 100) AS Percent_Rank_sal_PCT
FROM (
    SELECT
        category AS CAT,
        SUM(total) AS Sales, sum( QTY)  Total_QTY, sum(price) Total_price
    FROM
        data.order_sales
    GROUP BY
        category
)
```

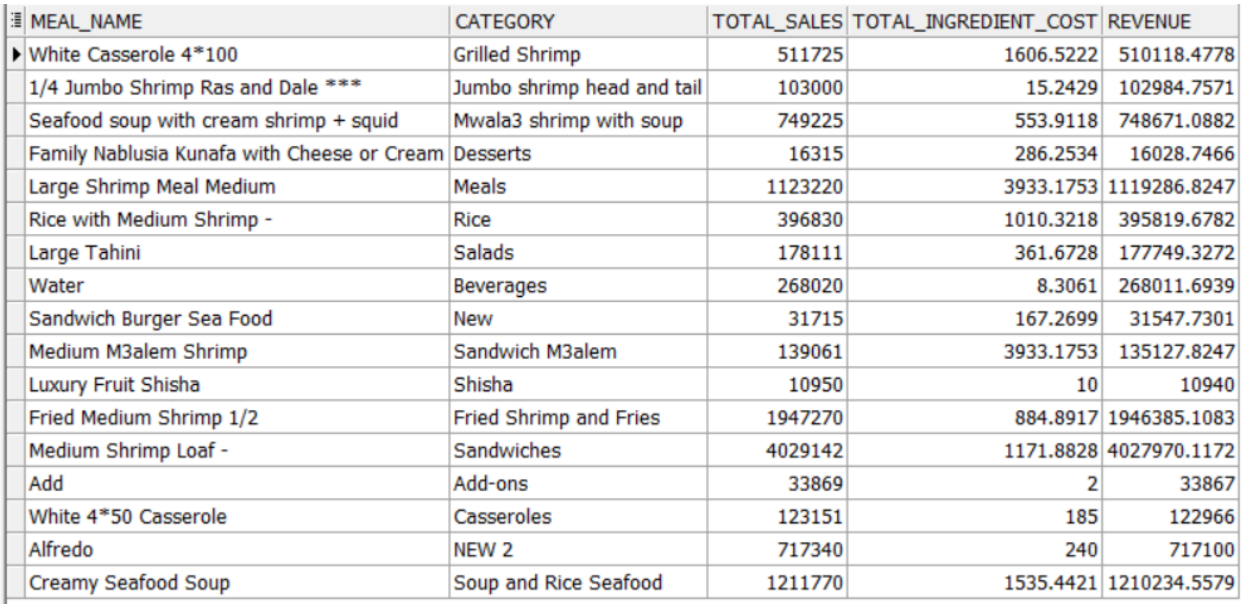
```
ORDER BY
  Percent_Rank_sal_PCT;
```



CAT	TOTAL_QTY	TOTAL_PRICE	SALES	PERCENT_RANK_SAL_PCT
Shisha	534	1050	20385	0
Add-ons	35141	101	45944	5
Desserts	1178	3875	58715	10
New	718	3065	61465	15
Staff	17534	504	79766	20
Special Meal	1320	1233	142741	25
Casseroles	729	6170	217531	30
Jumbo shrimp head and tail	1009	15120	333290	35
Sandwich M3alem	7766	5811	489684	40
Beverages	39805	9899	643842	45
Rice	26167	6280	1291910	50
Salads	51441	11450	1505113	55
Mwala3 shrimp with soup	99713	33318	1690068	60
NEW 2	19802	13165	2251260	65
Soup and Rice Seafood	24262	2780	2541420	70
Grilled Shrimp	27336	130482	5575828	75
Offers	30437	48870	6316557	80
Fish	12780130	10662	6788637	85
Meals	62057	93380	8040060	90
Sandwiches	214171	5057	8941352	95
Fried Shrimp and Fries	106350	178079	19592379	100

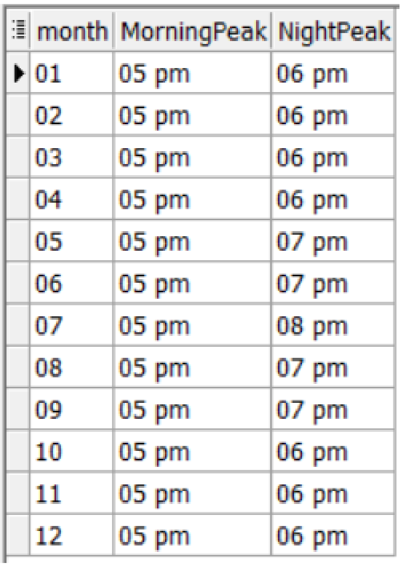
```
--------   the most seeling meal in each category and the revenu
WITH RankedMeals AS (
    SELECT
        Item_Name,
        Category,
        Total_Sales,
        ROW_NUMBER() OVER (PARTITION BY Category ORDER BY Total_Sales DESC) AS Rank
    FROM (
        SELECT
            Item_Name,
            Category,
            SUM(Total) AS Total_Sales
        FROM
            data.order_sales
        GROUP BY
            Item_Name,
            Category
    )
)
SELECT
```

```
    RM.Item_Name as Meal_Name,
    RM.Category,
    RM.Total_Sales,
    SUM(SI.Cost) AS Total_Ingredient_Cost,
    (RM.Total_Sales - SUM(SI.Cost)) AS Revenue
FROM
    RankedMeals RM
JOIN
    MealIngredients MI ON RM.Item_Name = MI.ItemName
JOIN
    ShrimpInventory SI ON MI.IngredientID = SI.code
WHERE
    RM.Rank = 1
GROUP BY
    RM.Item_Name,
    RM.Category,
    RM.Total_Sales;
```



MEAL_NAME	CATEGORY	TOTAL_SALES	TOTAL_INGREDIENT_COST	REVENUE
White Casserole 4*100	Grilled Shrimp	511725	1606.5222	510118.4778
1/4 Jumbo Shrimp Ras and Dale ***	Jumbo shrimp head and tail	103000	15.2429	102984.7571
Seafood soup with cream shrimp + squid	Mwala3 shrimp with soup	749225	553.9118	748671.0882
Family Nablusia Kunafa with Cheese or Cream	Desserts	16315	286.2534	16028.7466
Large Shrimp Meal Medium	Meals	1123220	3933.1753	1119286.8247
Rice with Medium Shrimp -	Rice	396830	1010.3218	395819.6782
Large Tahini	Salads	178111	361.6728	177749.3272
Water	Beverages	268020	8.3061	268011.6939
Sandwich Burger Sea Food	New	31715	167.2699	31547.7301
Medium M3alem Shrimp	Sandwich M3alem	139061	3933.1753	135127.8247
Luxury Fruit Shisha	Shisha	10950	10	10940
Fried Medium Shrimp 1/2	Fried Shrimp and Fries	1947270	884.8917	1946385.1083
Medium Shrimp Loaf -	Sandwiches	4029142	1171.8828	4027970.1172
Add	Add-ons	33869	2	33867
White 4*50 Casserole	Casseroles	123151	185	122966
Alfredo	NEW 2	717340	240	717100
Creamy Seafood Soup	Soup and Rice Seafood	1211770	1535.4421	1210234.5579

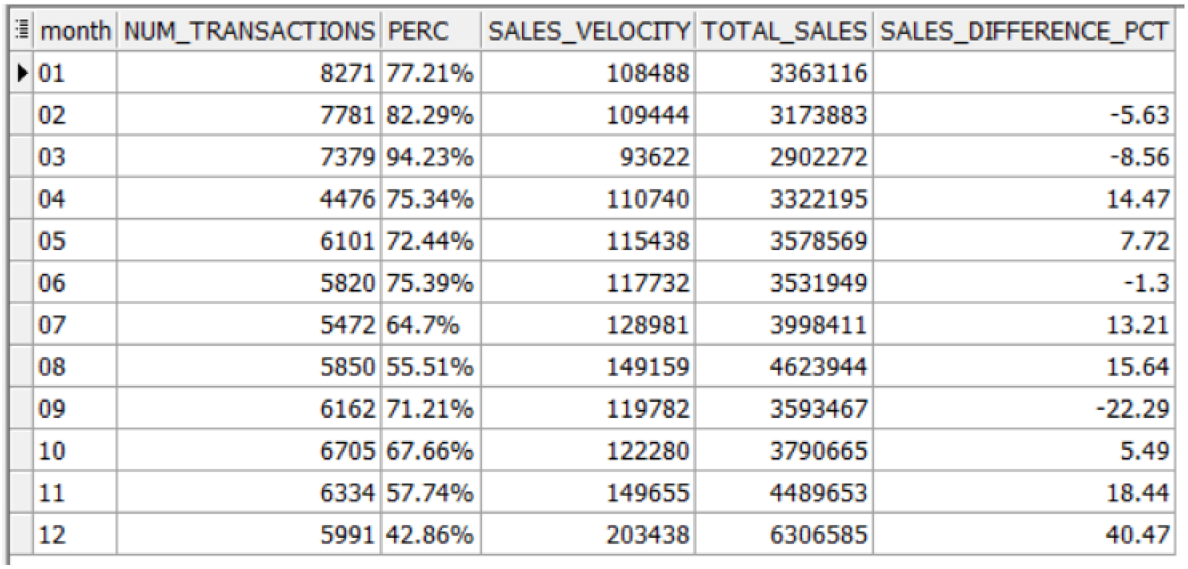
```
with shift_orders as (
    select "hour", "month",
        sum(case when emp_id in (11, 17) then 1 else 0 end) as "morning_count",
        sum(case when emp_id in (13, 14) then 1 else 0 end) as "night_count"
    from (
        select
                saleshеader.*,
            to_char(to_date(Time, 'HH:MI:SS am'), 'HH12 am') as "hour",
            to_char(Date_, 'MM') as "month"
        from
           data. salesheader
    )
    where
        emp_id in (11, 17, 13, 14)
    group by
        "hour", "month"
),
ranked_shift_orders as (
    select
        "month",
        max(case when "morning_count" = max_shift_1 then "hour" end) as "MorningPeak",
        max(case when "night_count" = max_shift_2 then "hour" end) as "NightPeak"
    from (
        select
            "hour",
            "month", "morning_count", "night_count",
            max("morning_count") over (partition by "month") as max_shift_1,
            max("night_count") over (partition by "month") as max_shift_2
        from
            shift_orders
    ) ranked_shift_orders
   group by
        "month"
)
select   "month",  "MorningPeak",  "NightPeak"
from  ranked_shift_orders
order by
    "month"
```



month	MorningPeak	NightPeak
01	05 pm	06 pm
02	05 pm	06 pm
03	05 pm	06 pm
04	05 pm	06 pm
05	05 pm	07 pm
06	05 pm	07 pm
07	05 pm	08 pm
08	05 pm	07 pm
09	05 pm	07 pm
10	05 pm	06 pm
11	05 pm	06 pm
12	05 pm	06 pm

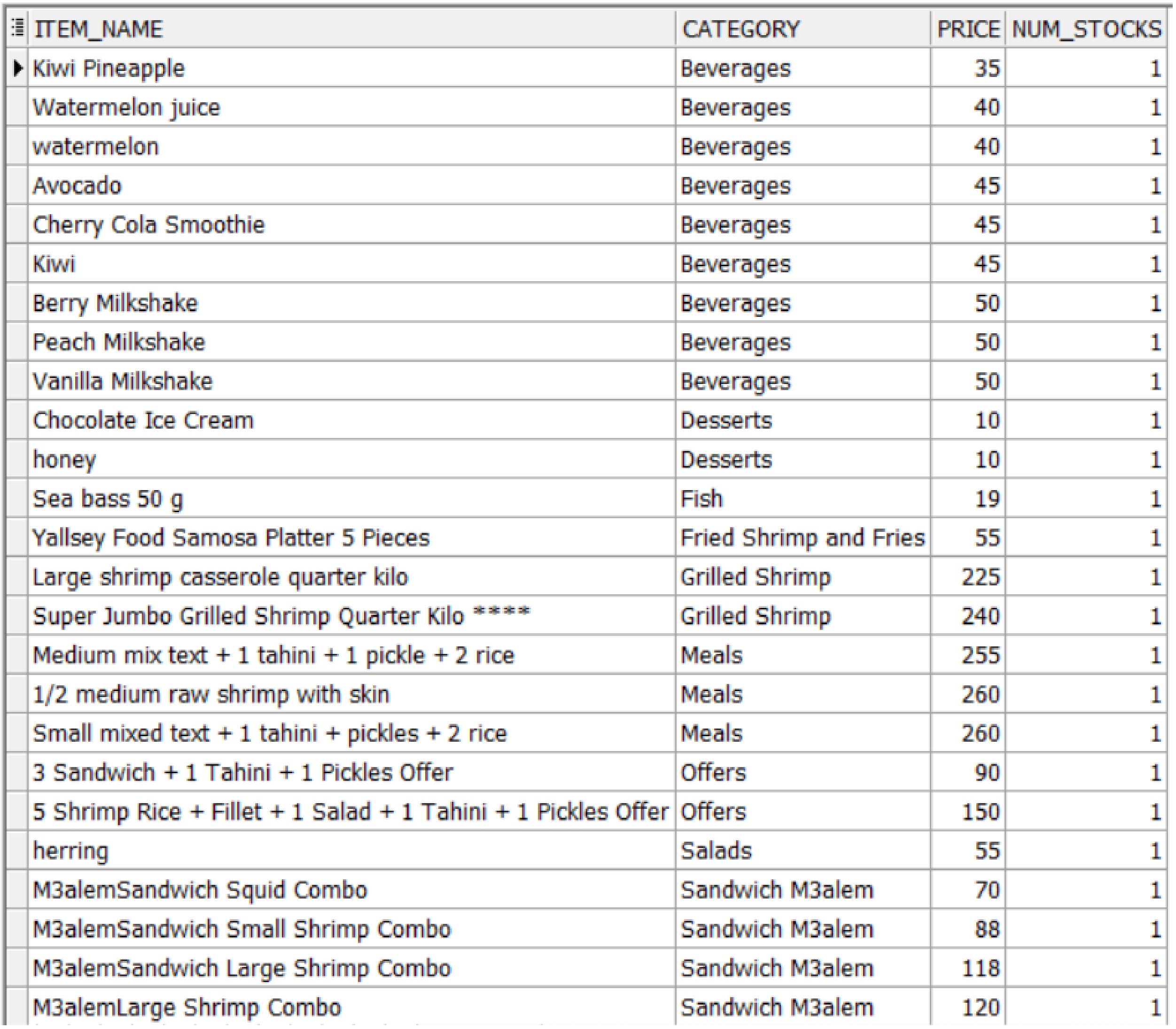
```
--------------------------------------------------------------------------------

with monthly_sales as (
    select
        to_char(Date_, 'MM') as "month",
        round(sum(grand_total)) as total_sales
    from
        data.salesheader
  group by
        to_char(DATE_, 'MM')
),
sales_velocity as (
    select
        m."month",
        m.total_sales,
        round((m.total_sales / to_number(to_char(last_day(to_date(m."month", 'MM')), 'DD')))) as velocity
    from
        monthly_sales m
),
total_sales_per_month as (
    select
        "month",
        total_sales,
        lag(total_sales) over (order by "month") as previous_month_sales
    from
        monthly_sales
),
trans_info as (
    select
        to_char(Date_, 'MM') as "month",
        grand_total,
        sum(grand_total) over (partition by to_char(Date_, 'MM') order by order_id) as runningtotal,
        row_number() over (partition by to_char(Date_, 'MM') order by order_id) as "count",
        count(order_id) over (partition by to_char(Date_, 'MM')) as "total"
    from
        data.salesheader
),
c as (
    select
        "month",
        "total",
        min("count") as num_transactions
    from
        trans_info
    where
        runningtotal >= 2600000 --monthly benchmark
    group by
        "month", "total"
)

select
    c."month",
    c.num_transactions,
    round((c.num_transactions / c."total") * 100, 2) || '%' as perc,
    v.velocity as sales_velocity,
    total.total_sales,
    round(((total.total_sales - total.previous_month_sales) / total.previous_month_sales) * 100, 2) as
sales_difference_pct
from
    c
join
    sales_velocity v on c."month" = v."month"
join
    total_sales_per_month total on c."month" = total."month"
```

```
order by
    c."month" asc
```



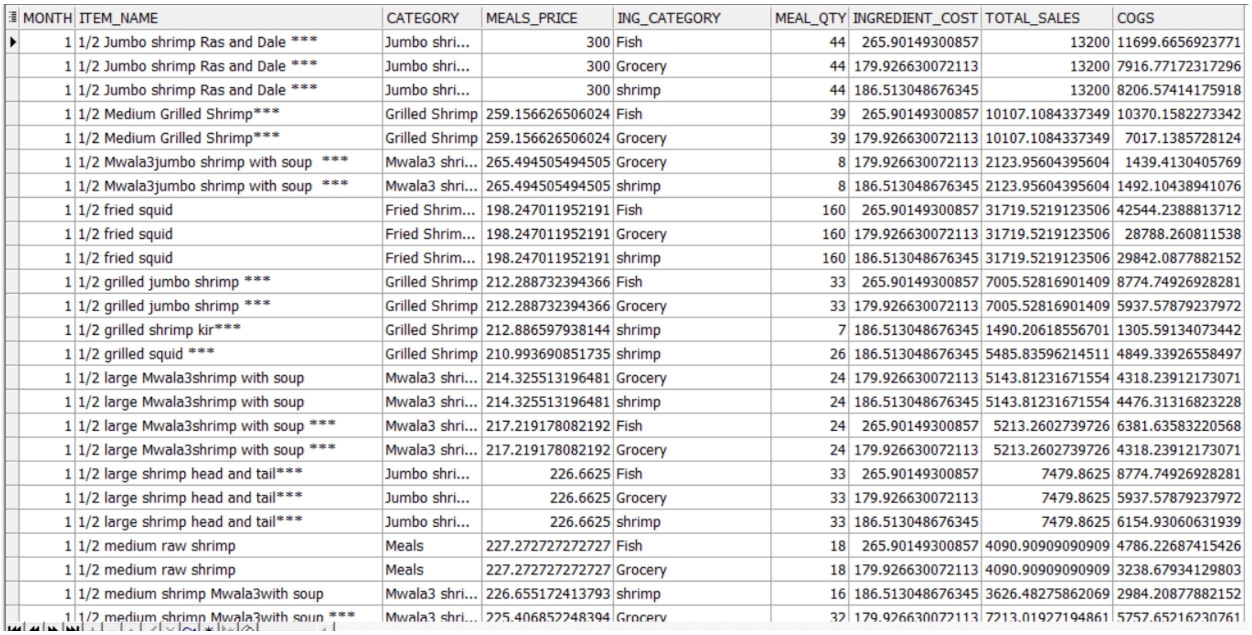
month	NUM_TRANSACTIONS	PERC	SALES_VELOCITY	TOTAL_SALES	SALES_DIFFERENCE_PCT
01	8271	77.21%	108488	3363116	
02	7781	82.29%	109444	3173883	-5.63
03	7379	94.23%	93622	2902272	-8.56
04	4476	75.34%	110740	3322195	14.47
05	6101	72.44%	115438	3578569	7.72
06	5820	75.39%	117732	3531949	-1.3
07	5472	64.7%	128981	3998411	13.21
08	5850	55.51%	149159	4623944	15.64
09	6162	71.21%	119782	3593467	-22.29
10	6705	67.66%	122280	3790665	5.49
11	6334	57.74%	149655	4489653	18.44
12	5991	42.86%	203438	6306585	40.47

```
  --items purchased only once
  SELECT  Item_Name, category, price,COUNT(*) AS Num_Stocks
    FROM (
        SELECT  Item_Name, price , category, SUM(QTY) OVER (PARTITION BY Item_Name) AS
Total_Quantity
        FROM order_sales
    )
    WHERE Total_Quantity = 1
  GROUP BY Item_Name,price, category order by category;
```



ITEM_NAME	CATEGORY	PRICE	NUM_STOCKS
Kiwi Pineapple	Beverages	35	1
Watermelon juice	Beverages	40	1
watermelon	Beverages	40	1
Avocado	Beverages	45	1
Cherry Cola Smoothie	Beverages	45	1
Kiwi	Beverages	45	1
Berry Milkshake	Beverages	50	1
Peach Milkshake	Beverages	50	1
Vanilla Milkshake	Beverages	50	1
Chocolate Ice Cream	Desserts	10	1
honey	Desserts	10	1
Sea bass 50 g	Fish	19	1
Yallsey Food Samosa Platter 5 Pieces	Fried Shrimp and Fries	55	1
Large shrimp casserole quarter kilo	Grilled Shrimp	225	1
Super Jumbo Grilled Shrimp Quarter Kilo ****	Grilled Shrimp	240	1
Medium mix text + 1 tahini + 1 pickle + 2 rice	Meals	255	1
1/2 medium raw shrimp with skin	Meals	260	1
Small mixed text + 1 tahini + pickles + 2 rice	Meals	260	1
3 Sandwich + 1 Tahini + 1 Pickles Offer	Offers	90	1
5 Shrimp Rice + Fillet + 1 Salad + 1 Tahini + 1 Pickles Offer	Offers	150	1
herring	Salads	55	1
M3alemSandwich Squid Combo	Sandwich M3alem	70	1
M3alemSandwich Small Shrimp Combo	Sandwich M3alem	88	1
M3alemSandwich Large Shrimp Combo	Sandwich M3alem	118	1
M3alemLarge Shrimp Combo	Sandwich M3alem	120	1

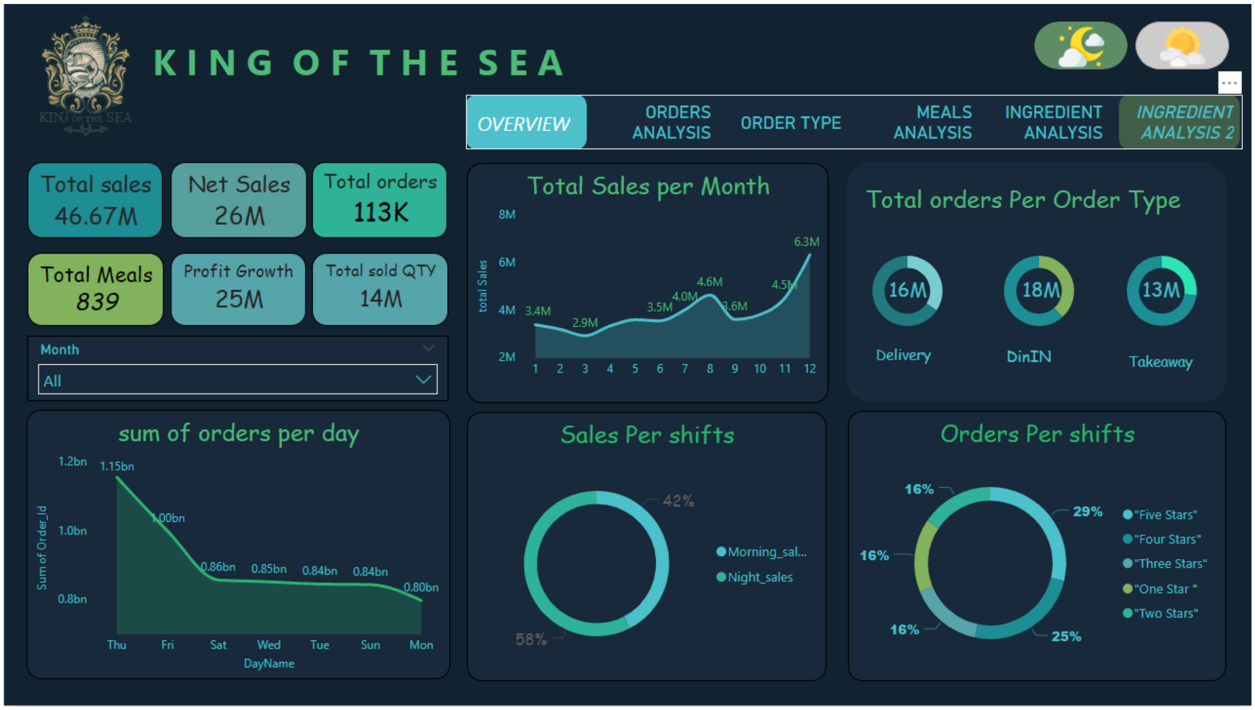
```
WITH cost_ingredient AS (
    SELECT
        RM.month,
        RM.item_name AS Item_name,
        RM.category AS Category,
        AVG(RM.price) OVER(ORDER BY RM.item_name) AS Meals_price,
        MI.ingrediant_category AS ING_Category,
        AVG(MI.cost) OVER(ORDER BY MI.ingrediant_category) AS Ingredient_cost,
        COUNT(RM.QTY) OVER(PARTITION BY RM.item_name) AS Meal_QTY
    FROM
        data.order_sales RM
    JOIN
        data.ingredient_fact MI ON RM.Item_Name = MI.Item_Name
)
SELECT
    month,
    Item_name,
    Category,
    Meals_price,
    ING_Category,
    Meal_QTY,
    Ingredient_cost,
    AVG(Meals_price * Meal_QTY) AS Total_sales,
    AVG(Meal_QTY * Ingredient_cost) AS COGS
FROM
    cost_ingredient
GROUP BY
    month,
    Item_name,
    Category,
    Meals_price,
    ING_Category,
    Ingredient_cost,
    Meal_QTY order by month;
```



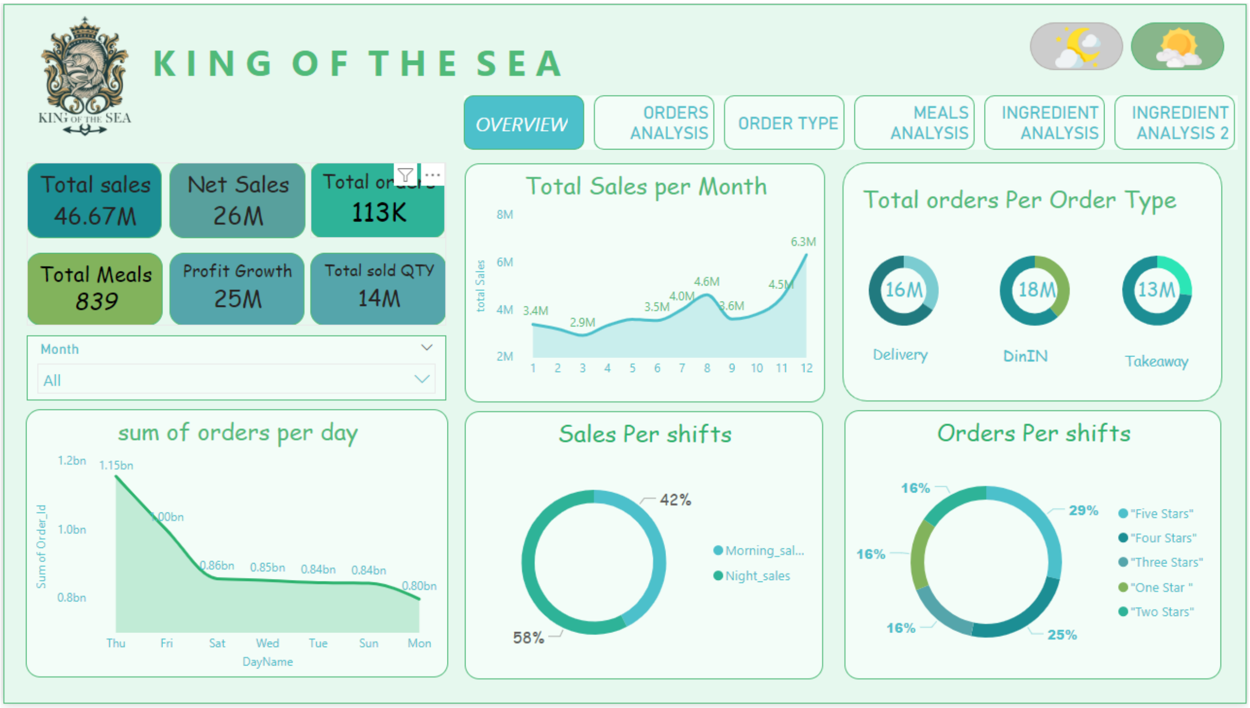
MONTH	ITEM_NAME	CATEGORY	MEALS_PRICE	ING_CATEGORY	MEAL_QTY	INGREDIENT_COST	TOTAL_SALES	COGS
1	1/2 Jumbo shrimp Ras and Dale ***	Jumbo shri...	300	Fish	44	265.90149300857	13200	11699.6656923771
1	1/2 Jumbo shrimp Ras and Dale ***	Jumbo shri...	300	Grocery	44	179.926630072113	13200	7916.77172317296
1	1/2 Jumbo shrimp Ras and Dale ***	Jumbo shri...	300	shrimp	44	186.513048676345	13200	8206.57414175918
1	1/2 Medium Grilled Shrimp***	Grilled Shrimp	259.156626506024	Fish	39	265.90149300857	10107.1084337349	10370.1582273342
1	1/2 Medium Grilled Shrimp***	Grilled Shrimp	259.156626506024	Grocery	39	179.926630072113	10107.1084337349	7017.1385728124
1	1/2 Mwala3jumbo shrimp with soup ***	Mwala3 shri...	265.494505494505	Grocery	8	179.926630072113	2123.95604395604	1439.4130405769
1	1/2 Mwala3jumbo shrimp with soup ***	Mwala3 shri...	265.494505494505	shrimp	8	186.513048676345	2123.95604395604	1492.10438941076
1	1/2 fried squid	Fried Shrim...	198.247011952191	Fish	160	265.90149300857	31719.5219123506	42544.2388813712
1	1/2 fried squid	Fried Shrim...	198.247011952191	Grocery	160	179.926630072113	31719.5219123506	28788.260811538
1	1/2 fried squid	Fried Shrim...	198.247011952191	shrimp	160	186.513048676345	31719.5219123506	29842.0877882152
1	1/2 grilled jumbo shrimp ***	Grilled Shrimp	212.288732394366	Fish	33	265.90149300857	7005.52816901409	8774.74926928281
1	1/2 grilled jumbo shrimp ***	Grilled Shrimp	212.288732394366	Grocery	33	179.926630072113	7005.52816901409	5937.57879237972
1	1/2 grilled shrimp kir***	Grilled Shrimp	212.886597938144	shrimp	7	186.513048676345	1490.20618556701	1305.59134073442
1	1/2 grilled squid ***	Grilled Shrimp	210.993690851735	shrimp	26	186.513048676345	5485.83596214511	4849.33926558497
1	1/2 large Mwala3shrimp with soup	Mwala3 shri...	214.325513196481	Grocery	24	179.926630072113	5143.81231671554	4318.23912173071
1	1/2 large Mwala3shrimp with soup	Mwala3 shri...	214.325513196481	shrimp	24	186.513048676345	5143.81231671554	4476.31316823228
1	1/2 large Mwala3shrimp with soup ***	Mwala3 shri...	217.219178082192	Fish	24	265.90149300857	5213.2602739726	6381.63583220568
1	1/2 large Mwala3shrimp with soup ***	Mwala3 shri...	217.219178082192	Grocery	24	179.926630072113	5213.2602739726	4318.23912173071
1	1/2 large shrimp head and tail***	Jumbo shri...	226.6625	Fish	33	265.90149300857	7479.8625	8774.74926928281
1	1/2 large shrimp head and tail***	Jumbo shri...	226.6625	Grocery	33	179.926630072113	7479.8625	5937.57879237972
1	1/2 large shrimp head and tail***	Jumbo shri...	226.6625	shrimp	33	186.513048676345	7479.8625	6154.93060631939
1	1/2 medium raw shrimp	Meals	227.272727272727	Fish	18	265.90149300857	4090.90909090909	4786.22687415426
1	1/2 medium raw shrimp	Meals	227.272727272727	Grocery	18	179.926630072113	4090.90909090909	3238.67934129803
1	1/2 medium shrimp Mwala3with soup	Mwala3 shri...	226.655172413793	shrimp	16	186.513048676345	3626.48275862069	2984.20877882152
1	1/2 medium shrimp Mwala3with soup ***	Mwala3 shri...	225.406852248394	Grocery	32	179.926630072113	7213.01927194861	5757.65216230761

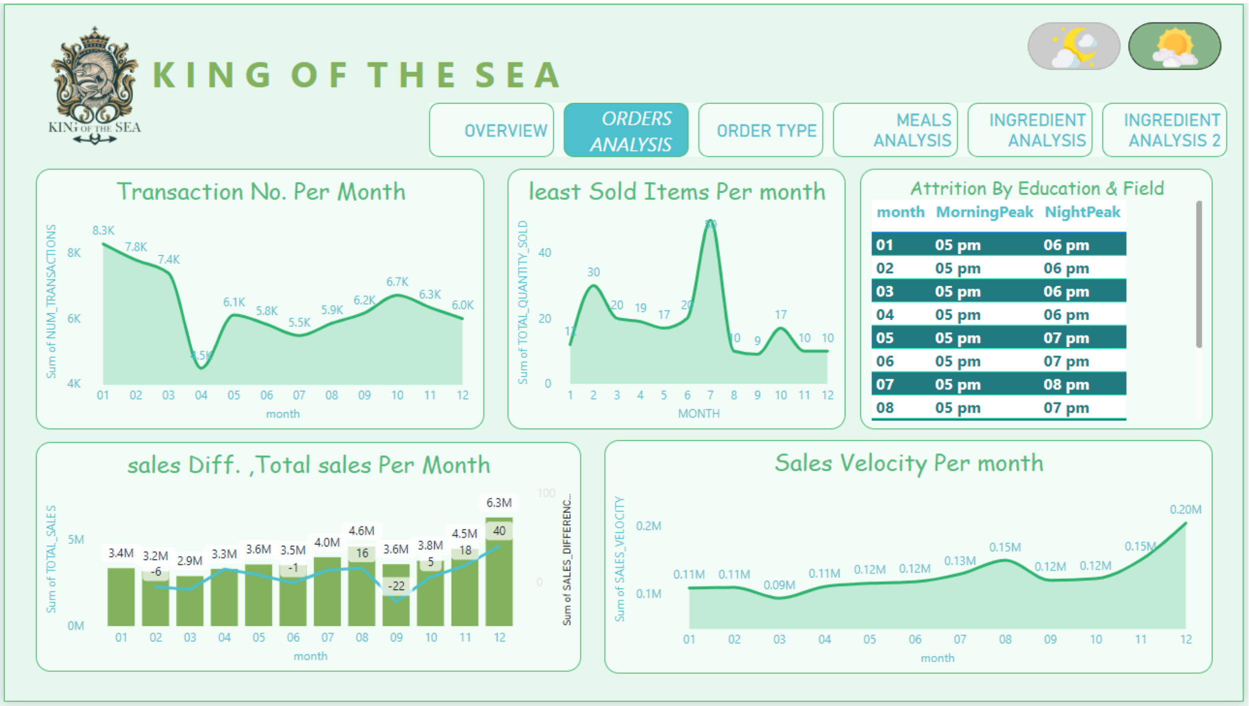
7 DASHBOARD:

Dark Mode

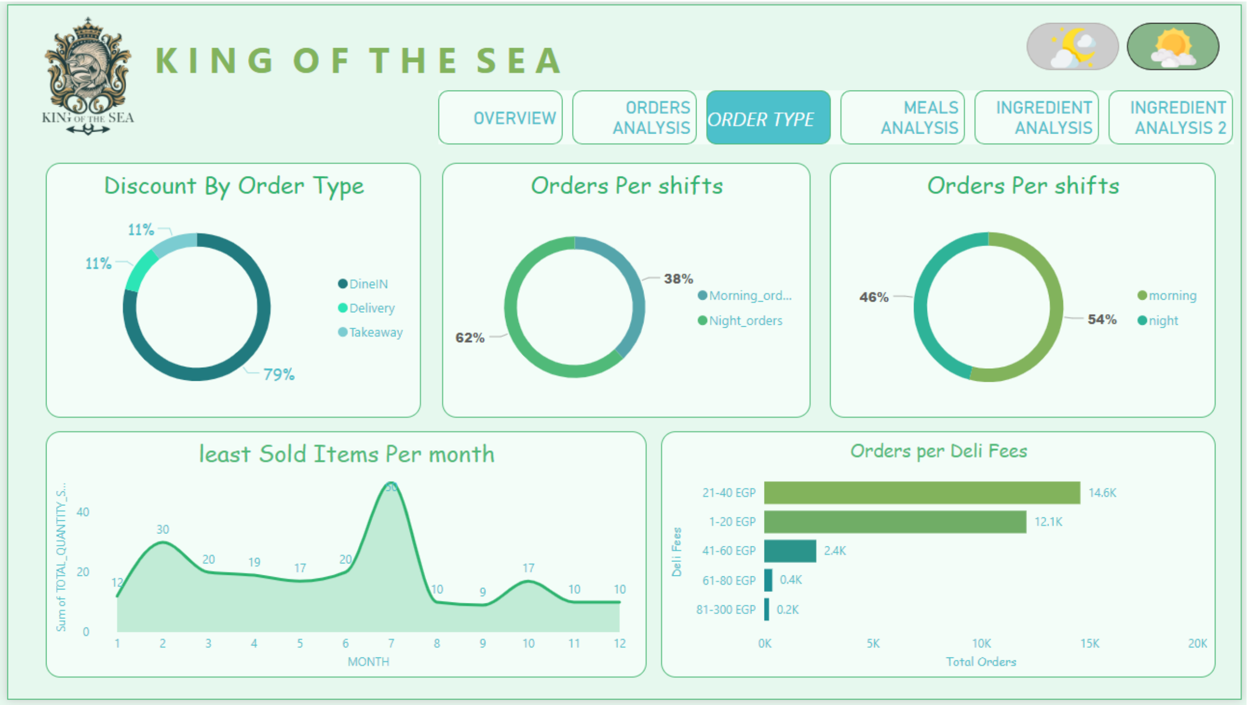


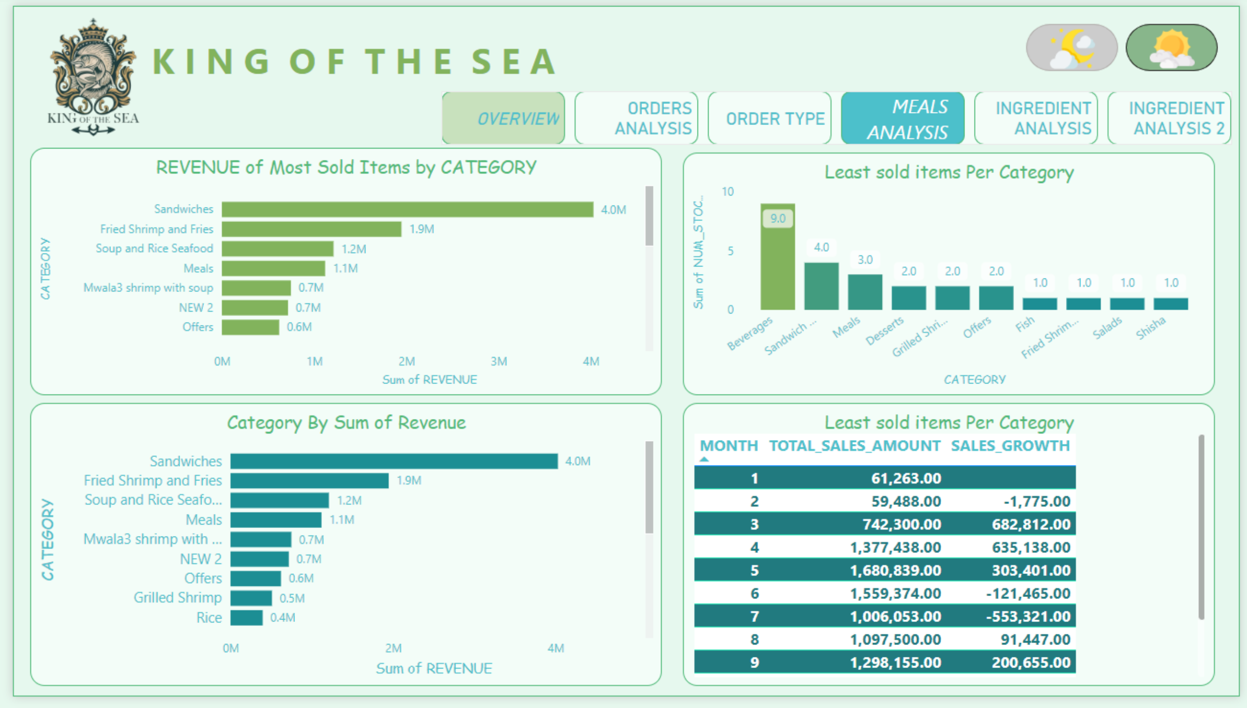
Light Mode



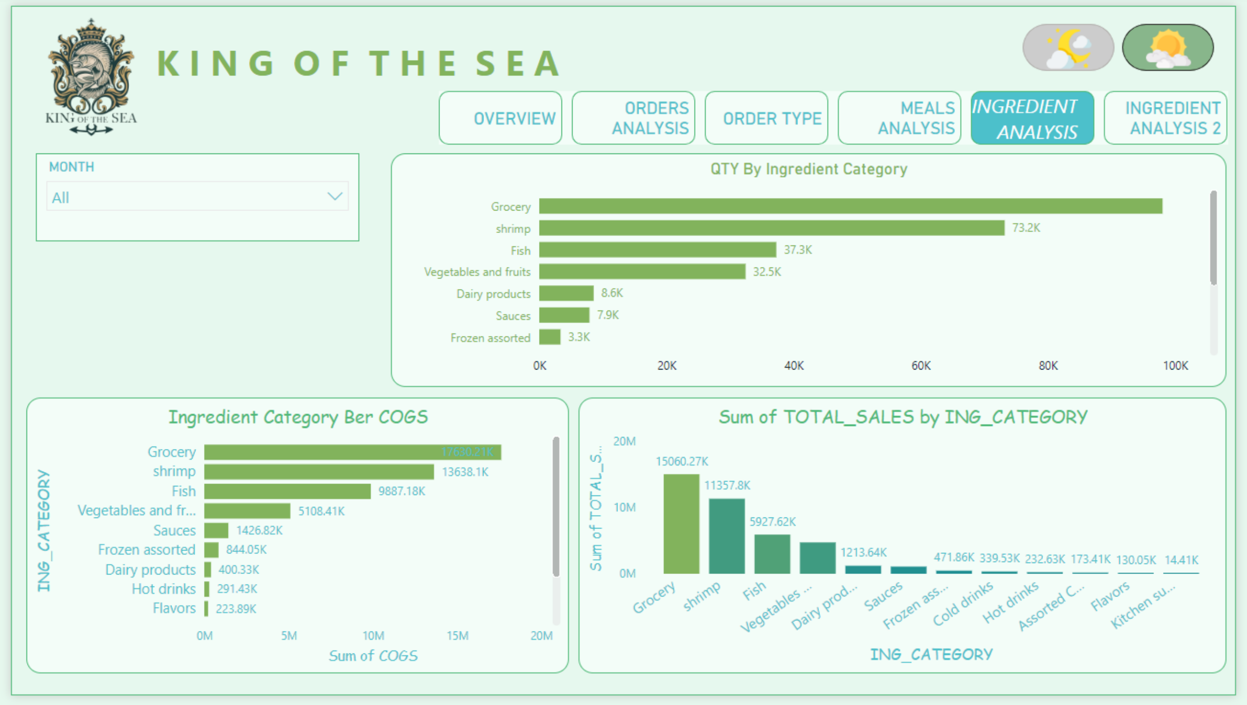


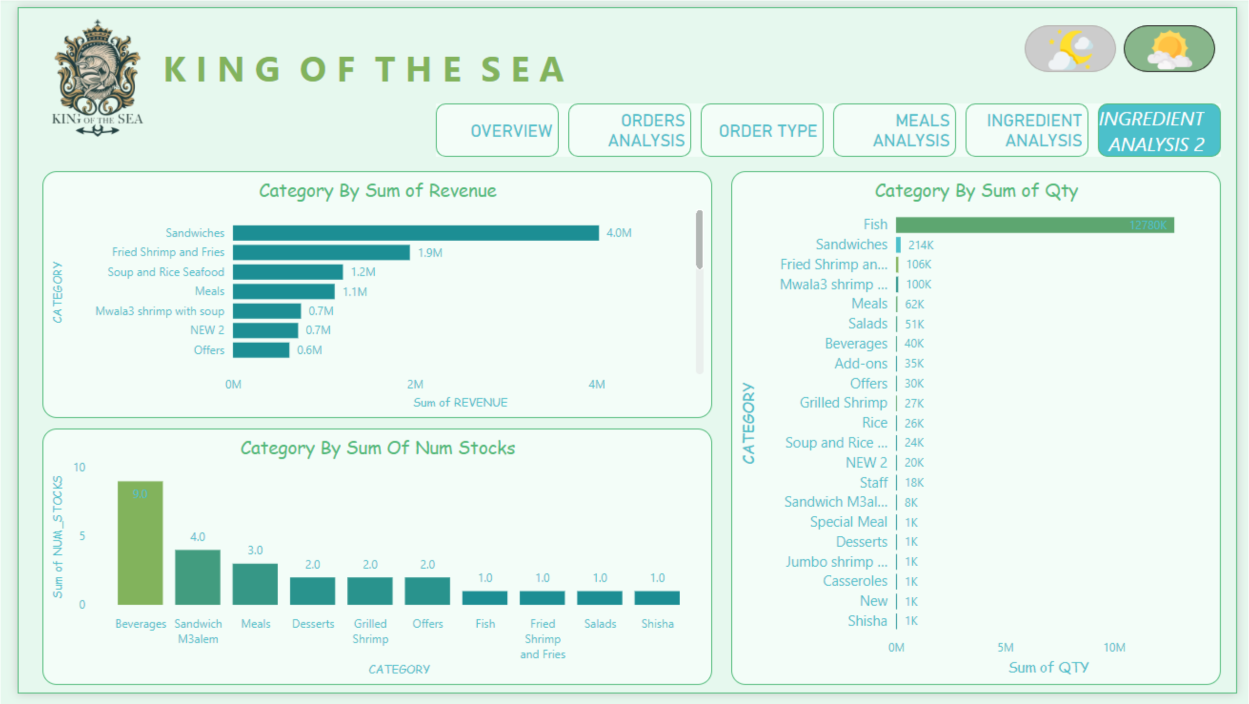
month	MorningPeak	NightPeak
01	05 pm	06 pm
02	05 pm	06 pm
03	05 pm	06 pm
04	05 pm	06 pm
05	05 pm	07 pm
06	05 pm	07 pm
07	05 pm	08 pm
08	05 pm	07 pm





MONTH	TOTAL_SALES_AMOUNT	SALES_GROWTH
1	61,263.00	
2	59,488.00	-1,775.00
3	742,300.00	682,812.00
4	1,377,438.00	635,138.00
5	1,680,839.00	303,401.00
6	1,559,374.00	-121,465.00
7	1,006,053.00	-553,321.00
8	1,097,500.00	91,447.00
9	1,298,155.00	200,655.00





8 SQL QUERIES:

```
CREATE TABLE ATTENDANCE_FACT

(

   EMP_ID   NUMBER(38) NOT NULL,

   DATE_    DATE NOT NULL,

   INTIME   TIMESTAMP,

   OUTTIME  TIMESTAMP,

   DAY_NAME VARCHAR2(20),

   CONSTRAINT pk_attendance_fact PRIMARY KEY (EMP_ID, DATE_),

   CONSTRAINT fk_attendance_emp FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE(EMP_ID)

);
```

```
CREATE TABLE EMP_DIM
```

```
(
   SHIFT              VARCHAR2(50),
   START_             TIMESTAMP,
   END_               TIMESTAMP,
   TOTAL_HOURS        NUMBER(6,2),
   EMPLOYEE_ID        NUMBER(38) PRIMARY KEY,
   FIRST_NAME         NVARCHAR2(50),
   LAST_NAME          NVARCHAR2(50),
   AGE                NUMBER(3),
   PHONE_NUMBER       NVARCHAR2(20),
   JOB_TITLE          NVARCHAR2(50),
   OFFDAY             NVARCHAR2(20),
   PERFORMANCE_EVALUATION NUMBER(3,2),


   -- Constraints
   CONSTRAINT emp_dim_age_check CHECK (AGE >= 18 AND AGE <= 100),
   CONSTRAINT emp_dim_perf_eval_check CHECK (PERFORMANCE_EVALUATION >= 0 AND
PERFORMANCE_EVALUATION <= 5),
   CONSTRAINT emp_dim_start_end_check CHECK (START_ <= END_),
   CONSTRAINT emp_dim_shift_not_null CHECK (SHIFT IS NOT NULL),
   CONSTRAINT emp_dim_phone_format CHECK (REGEXP_LIKE(PHONE_NUMBER,
'^(\+\d{1,2}\s)?\(?\d{3}\)?[\s.-]?\d{3}[\s.-]?\d{4}$'))
);




CREATE TABLE INGREDIENT_DIM
```

```
(
    MEAL_ING_ID             NVARCHAR2(26) PRIMARY KEY,
    ITEM_NAME               NVARCHAR2(128),
    ITEM_ID                 NUMBER(38),
    CATEGORY                NVARCHAR2(50),
    INGREDIENT_ID           NUMBER(38),
    INGREDIENT_CATEGORY     NVARCHAR2(50),
    INGREDIENT_CATEGORY_CODE NUMBER(10),

    -- Constraints
    CONSTRAINT unique_item_id UNIQUE (ITEM_ID),
    CONSTRAINT unique_ingr_id UNIQUE (INGREDIENT_ID),
    CONSTRAINT chk_ingr_category CHECK (INGREDIENT_CATEGORY IN ('Vegetable', 'Fruit', 'Meat',
'Dairy', 'Grain', 'Other')),
    CONSTRAINT chk_ingr_category_code CHECK (INGREDIENT_CATEGORY_CODE >= 0),
    CONSTRAINT fk_ingr_category FOREIGN KEY (INGREDIENT_CATEGORY_CODE) REFERENCES
CATEGORY_DIM(CATEGORY_CODE)
);
```

```
CREATE TABLE ingredient_FACT_FINAL
(
    INVENTORYID             NVARCHAR2(26),
    MEAL_ING_ID             NVARCHAR2(26),
    ITEM_NAME              NVARCHAR2(128),

    ITEM_ID                NUMBER(38),

    PRICE                  NUMBER(12,2),

    CATEGORY               NVARCHAR2(50),

    ITEM                   NVARCHAR2(128),

    CODE                   NUMBER(38),

    ITEMGROUP              NVARCHAR2(31),

    UNIT                   NVARCHAR2(26),

    ITEMGROUPCODE          NUMBER(38),

    QUANTITY               NUMBER(38,3),

    COST                   NUMBER(12,2),

    AMOUNT                 NUMBER(12,4),

    ADDITION               NUMBER(38,3),

    SUBTRACTION            NUMBER(38,3),

    STOCK                  NVARCHAR2(26),

    INGREDIENT_ID          NUMBER(38),

    INGREDIENT_CATEGORY    NVARCHAR2(50),

    INGREDIENT_CATEGORY_CODE NUMBER(10),


    -- Constraints

    CONSTRAINT pk_ingredient_fact_final PRIMARY KEY (INVENTORYID),

    CONSTRAINT fk_ingredient_fact_final_meal_ing FOREIGN KEY (MEAL_ING_ID) REFERENCES
INGREDIENT_DIM(MEAL_ING_ID),

    CONSTRAINT fk_ingredient_fact_final_item_id FOREIGN KEY (ITEM_ID) REFERENCES
INGREDIENT_DIM(ITEM_ID),

    CONSTRAINT fk_ingredient_fact_final_ingr_id FOREIGN KEY (INGREDIENT_ID) REFERENCES
INGREDIENT_DIM(INGREDIENT_ID),

    CONSTRAINT fk_ingredient_fact_final_ingr_cat_code FOREIGN KEY (INGREDIENT_CATEGORY_CODE)
REFERENCES INGREDIENT_DIM(INGREDIENT_CATEGORY_CODE)

);
```

```
CREATE TABLE inventory_DIM20
(
    INVENTORYID   NVARCHAR2(26) PRIMARY KEY,
    ITEM          NVARCHAR2(128),
    CODE          NUMBER(38),
    ITEMGROUP     NVARCHAR2(31),
    UNIT          NVARCHAR2(26),
    ITEMGROUPCODE NUMBER(38),
    STOCK         NVARCHAR2(26),
    SUPPLIER_ID   NUMBER(38),
    NAME          NVARCHAR2(50),
    CONTACT_INFO  NVARCHAR2(50),
    SUPPLIES_TYPE NVARCHAR2(128),

    -- Constraints
    CONSTRAINT inventory_dim20_supplier_fk FOREIGN KEY (SUPPLIER_ID) REFERENCES
SUPPLIER_DIM(SUPPLIER_ID),
    CONSTRAINT inventory_dim20_stock_check CHECK (STOCK IN ('In stock', 'Out of stock')),
    CONSTRAINT inventory_dim20_code_unique UNIQUE (CODE)
);



CREATE TABLE Iteam_sales_FACT1
(
    ITEM_NAME NVARCHAR2(256),
    CATEGORY  NVARCHAR2(128),
    QTY       NUMBER(38),
    PRICE     NUMBER(12, 2),
    TOTAL     NUMBER(12, 2),
    DATE_     DATE,
    MONTH     NUMBER(2),
    YEAR      NUMBER(4),
    ITEM_ID   NUMBER(38),

    -- Constraints
    CONSTRAINT item_sales_fact1_qty_check CHECK (QTY >= 0),
    CONSTRAINT item_sales_fact1_price_check CHECK (PRICE >= 0),
    CONSTRAINT item_sales_fact1_total_check CHECK (TOTAL >= 0),
    CONSTRAINT item_sales_fact1_date_check CHECK (EXTRACT(YEAR FROM DATE_) = YEAR),
    CONSTRAINT item_sales_fact1_month_check CHECK (MONTH >= 1 AND MONTH <= 12), --
    CONSTRAINT item_sales_fact1_item_id_fk FOREIGN KEY (ITEM_ID) REFERENCES
inventory_DIM20(ITEM_ID)
);



CREATE TABLE MENU_DIM
(
    ITEM_NAME    NVARCHAR2(128),
    ITEM_ID      NUMBER(38) PRIMARY KEY,
    PRICE        NUMBER(12, 2),
    CATEGORY     NVARCHAR2(50),
    AVAILABILITY NVARCHAR2(20),
    DESCRIPTION  NVARCHAR2(256),

    -- Constraints
    CONSTRAINT menu_dim_price_check CHECK (PRICE >= 0),
    CONSTRAINT menu_dim_availability_check CHECK (AVAILABILITY IN (Yes, 'No')),
    CONSTRAINT menu_dim_item_name_unique UNIQUE (ITEM_NAME)
);




CREATE TABLE ORDER_HEADER_FACT
(
    ORDER_ID    NUMBER(38) PRIMARY KEY,
    DATE_       TIMESTAMP,
    TIME        NVARCHAR2(26),
    ORDER_TYPE  NVARCHAR2(26),
    SUBTOTAL    NUMBER(12, 2),
    DISCOUNT    NUMBER(5, 1),
    TAX         NUMBER(8, 2),
    SERVICE     NUMBER(8, 2),
    DELI_FEES   NUMBER(12),
    GRAND_TOTAL NUMBER(12, 2),
    EMP_ID      NUMBER(38),
    STATUS      NVARCHAR2(26),

    -- Constraints
```

```
  CONSTRAINT order_header_fact_subtotal_check CHECK (SUBTOTAL >= 0),

  CONSTRAINT order_header_fact_discount_check CHECK (DISCOUNT >= 0 AND DISCOUNT <= 100),

  CONSTRAINT order_header_fact_tax_check CHECK (TAX >= 0),

  CONSTRAINT order_header_fact_service_check CHECK (SERVICE >= 0),

  CONSTRAINT order_header_fact_deli_fees_check CHECK (DELI_FEES >= 0),

  CONSTRAINT order_header_fact_grand_total_check CHECK (GRAND_TOTAL >= 0),

  CONSTRAINT order_header_fact_status_check CHECK (STATUS IN ('Pending', 'Completed',
'Cancelled'))

);





CREATE TABLE RESERVATION_DIM1

(

  RESERVATION_ID          NUMBER(38) PRIMARY KEY,

  RESERVATION_NAME        NVARCHAR2(50),

  DATE_                   TIMESTAMP,

  TIME                    NVARCHAR2(26),

  TABLE_NAME              NVARCHAR2(50),

  SPECIAL_RESERVATION_REQUEST NVARCHAR2(256),

  PARTY_SIZE              NUMBER(38),

  ORDER_ID                NUMBER(38),


  -- Constraints

  CONSTRAINT reservation_dim1_party_size_check CHECK (PARTY_SIZE > 0),

  CONSTRAINT reservation_dim1_order_fk FOREIGN KEY (ORDER_ID) REFERENCES
ORDER_HEADER_FACT(ORDER_ID)

);
```

```
CREATE TABLE SEATINGS_DIM
(
   TABLE_ID     NUMBER(38) PRIMARY KEY,
   TABLE_NUMBER NVARCHAR2(26),
   CAPACITY     NUMBER(38),
   AVAILABILITY NVARCHAR2(20),

   -- Constraints
   CONSTRAINT seatings_dim_capacity_check CHECK (CAPACITY > 0),
   CONSTRAINT seatings_dim_availability_check CHECK (AVAILABILITY IN ('Available', 'Occupied',
'Reserved'))
);
```

```
CREATE TABLE EMP_DIM
(
   SHIFT               nvarchar2(26),
   START_              nvarchar2(26),
   END                 nvarchar2(26),
   TOTAL_HOURS           number(38),
   EMPLOYEE_ID           number(38),
   FIRST_NAME           nvarchar2(26),
   LAST_NAME            nvarchar2(26),
   AGE                  number(38),
   PHONE_NUMBER         nvarchar2(26),
   JOB_TITLE            nvarchar2(26),
   OFFDAY               nvarchar2(26),
   PREFORMANCE_EVALUATION number(38)
);



CREATE TABLE INGREDIANT_DIM
(
   MEAL_ING_ID            nvarchar2(26),
   ITEM_NAME              nvarchar2(128),
   ITEM_ID               number(38),
   CATEGORY              nvarchar2(26),
   INGREDIANT_ID          number(38),
   INGREDIANT_CATEGORY     nvarchar2(26),
   INGREDIANT_CATEGORY_CODE number(38)
);
```

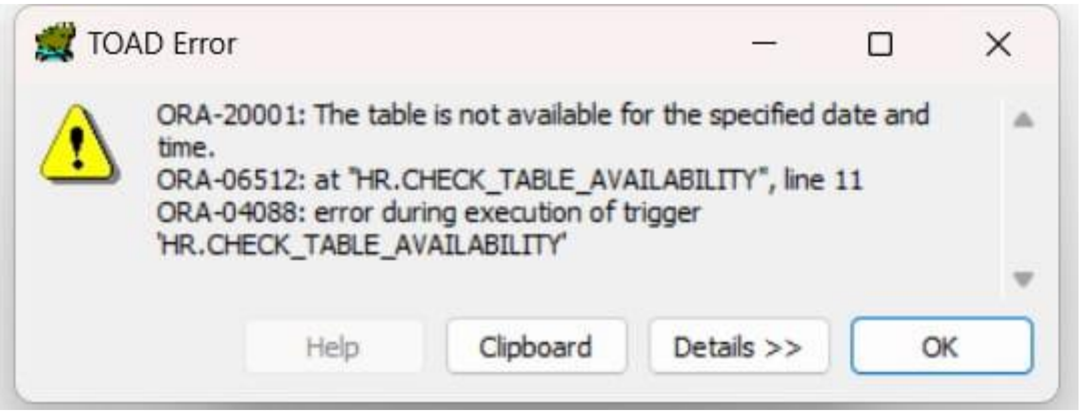
```
CREATE TABLE ingredient_FACT_FINAL
(
   INVENTORYID             nvarchar2(26),
   MEAL_ING_ID             nvarchar2(26),
   ITEM_NAME               nvarchar2(128),
   ITEM_ID                 number(38),
   PRICE                   number(38),
   CATEGORY                nvarchar2(26),
   ITEM                    nvarchar2(128),
   CODE                    number(38),
   ITEMGROUP               nvarchar2(31),
   UNIT                    nvarchar2(26),
   ITEMGROUPCODE           number(38),
   QUANTITY                number(38,3),
   COST                    number(38,4),
   AMOUNT                  number(38,4),
   ADDITION                number(38,3),
   SUBTRACTION             number(38,3),
   STOCK                   nvarchar2(26),
   INGREDIANT_ID           number(38),
   INGREDIANT_CATEGORY     nvarchar2(26),
   INGREDIANT_CATEGORY_CODE number(38)
);



CREATE TABLE inventory_DIM20
(
   INVENTORYID   nvarchar2(26),
   ITEM         nvarchar2(128),
   CODE         number(38),
   ITEMGROUP    nvarchar2(31),
   UNIT         nvarchar2(26),
   ITEMGROUPCODE number(38),
   STOCK        nvarchar2(26),
   SUPPLIER_ID   number(38),
   NAME         nvarchar2(26),
   CONTACT_INFO  nvarchar2(26),
   SUPPLIES_TYPE nvarchar2(128)
);



CREATE TABLE Iteam_sales_FACT1
(
   ITEM_NAME nvarchar2(256),
   CATEGORY  nvarchar2(128),
   QTY      number(38),
   PRICE    number(38),
   TOTAL    number(38),
   DATE_    timestamp(9),
   MONTH    number(38),
   YEAR     number(38),
   ITEM_ID   number(38)
);
```

```
CREATE TABLE MENU_DIM
(
   ITEM_NAME    nvarchar2(128),
   ITEM_ID      number(38),
   PRICE        number(38),
   CATEGORY     nvarchar2(26),
   AVAILABILITY nvarchar2(26),
   DESCRIPTION  nvarchar2(256)
);
```

```
CREATE TABLE ORDER_HEADER_FACT
(
   ORDER_ID    number(38),
   DATE_       timestamp(9),
   TIME        nvarchar2(26),
   ORDER_TYPE  nvarchar2(26),
   SUBTOTAL    number(38),
   DISCOUNT    number(38,1),
   TAX         number(38,2),
   SERVICE     number(38,2),
   DELI_FEES   number(38),
   GRAND_TOTAL number(38,2),
   EMP_ID      number(38),
   STATUS      nvarchar2(26)
);
```

```
CREATE TABLE RESERVATION_DIM1

(

   RESERVATION_ID             number(38),

   RESERVATION_NAME           nvarchar2(26),

   DATE_                 timestamp(9),

   TIME                  nvarchar2(26),

   TABLE_NAME                nvarchar2(26),

   SPECIAL_RESERVATION_REQUEST nvarchar2(128),

   PARTY_SIZE                number(38),

   ORDER_ID                  number(38)

);




CREATE TABLE SEATINGS_DIM

(

   TABLE_ID     number(38),

   TABLE_NUMBER nvarchar2(26),

   CAPACITY     number(38),

   AVAILABILITY nvarchar2(26)

);


CREATE OR REPLACE TRIGGER check_table_availability

BEFORE INSERT ON reservations_dim

FOR EACH ROW
```

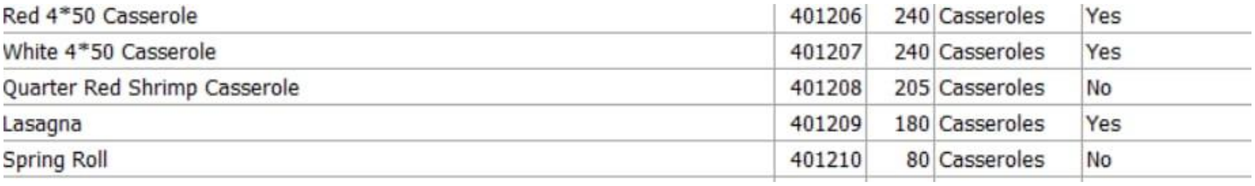
```
DECLARE
   v_availability VARCHAR2(20);
BEGIN
   -- Get the availability of the table
   SELECT t_Availability INTO v_availability
   FROM seatings_dim
   WHERE Table_number = :NEW.Table_name;

   -- Check if the table is available
   IF v_availability = 'Not avaliable' THEN
      RAISE_APPLICATION_ERROR(-20001, 'The table is not available for the specified date and time.');
   END IF;
END;
/
show errors
```



```
ORA-20001: The table is not available for the specified date and
time.
ORA-06512: at "HR.CHECK_TABLE_AVAILABILITY", line 11
ORA-04088: error during execution of trigger
'HR.CHECK_TABLE_AVAILABILITY'
```

```
CREATE OR REPLACE PROCEDURE check_meal_available IS
   CURSOR item_ids_cur IS
      SELECT DISTINCT item_id FROM inventory_fact;
   v_count NUMBER;
BEGIN
   -- Loop through each distinct item_id
```

```
  FOR item_rec IN item_ids_cur LOOP
    -- For each item_id, loop through its ingredients
    FOR ingredient_rec IN (SELECT ingrediant_id, quantity FROM inventory_fact WHERE item_id = item_rec.item_id) LOOP
      -- Check if the quantity is 0
      IF ingredient_rec.quantity = 0 THEN
        update menu_dim
        set m_availability='No'
        where item_id=item_rec.item_id;


      END IF;
    END LOOP;
  END LOOP;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No data found.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/

/

begin
check_meal_available;
end;
```



Red 4*50 Casserole	401206	240	Casseroles	Yes
White 4*50 Casserole	401207	240	Casseroles	Yes
Quarter Red Shrimp Casserole	401208	205	Casseroles	No
Lasagna	401209	180	Casseroles	Yes
Spring Roll	401210	80	Casseroles	No

```
CREATE OR REPLACE PROCEDURE notify_low_stock AS
  v_threshold NUMBER;
BEGIN
  -- Calculate the threshold as 50% of the quantity
  v_threshold := .5; -- Default value for the threshold, change as needed

  -- Query to find items with low stock levels
  FOR low_stock_rec IN (
    SELECT ITEM_NAME, QUANTITY, CATEGORY
    FROM inventory_fact
    WHERE QUANTITY < v_threshold * quantity
  ) LOOP
    -- Send notification for low stock item
    -- You can implement this part based on your notification mechanism
    DBMS_OUTPUT.PUT_LINE('Low stock for Item: ' || low_stock_rec.ITEM_NAME || ', Category: ' || low_stock_rec.CATEGORY || ', Quantity: ' || low_stock_rec.QUANTITY);
  END LOOP;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No items found with low stock levels.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

```
begin
notify_low_stock;
end;
```