BAIT3003 Data Warehouse

Assignment

2024

Programme : RDS2S3

Tutorial Group : 3

Date Submitted to Tutor : 15 September 2024

Team Members:

No	Student Name	Student ID
1.	Eugene Tan Yu Xian	22WMR14288
2.	Joash Alwinn Voon Dirui	22WMR14084
3.	Loh Jia Shou	22WMR13596
4.	Ng Hong Han	22WMR14232
5.	Vithiya Saraswathi a/p Sockalingam	22WMR13649

Create a data warehouse with the use of the Star-schema modelling technique.
The data warehouse should be based on your previous/current assignment of a database system, specifically from, BACS3183 Advanced database or BACS1053 Database Management course assignment.

Provide a brief write-up of the system.

Complete your assignment by documenting a report with the following:

1. Design a data warehouse based on the star-schema for management to use.
    a. Logical design. (5 marks)
    b. Physical design. (15 marks)
2. Show all the ETL processes necessary to populate the data warehouse.
    a. The initial loading of the data warehouse. (20 marks)
    b. Subsequent loading of the data warehouse. (20 marks)
3. Produce THREE(3) Business Analytics reports each to assist PSMS management to be more competitive. (30 marks) (Analysis of the various dimensions, across dimensions, etc.)
4. Present all the task in a proper report format (10 marks)

Total:100 marks

Due Date: Sunday 15 September 2024, 5 pm
(Presentation in Week 12/13)

BAIT3003 Data Warehouse Technology
Assignment Assessment Form

Task No.	Task Descriptions	Weightage	Criteria	Ratings	Marks	CLO
1	Design of Data warehouse (logical design)	5%	• Include the relevant dimensions. • Include the correct measures in the fact table.	• Excellent (5) • Good (4) • Moderate (2-3) • Poor (0-1)		1
	Design of Data warehouse (physical design)	15%	• Create TABLE statements • Appropriate data types and size of attributes • Proper Integrity constraints	• Excellent (13-15) • Good (10-12) • Moderate (6-9) • Poor (0-5)		1
2	ETL (initial loading)	20%	• VIEWS, SELECT,INSERT,PROCEDURES for each of the dimensions and fact table. • Variety of techniques necessary to achieve the correct data loading	• Excellent (18-20) • Good (14-17) • Moderate (9-13) • Poor (0-8)		1
	ETL (subsequent loading)	20%	• VIEWS, SELECT,INSERT,PROCEDURES for each of the dimensions and fact table. • Logic to scrub dirty data	• Excellent (18-20) • Good (15-17) • Moderate		1

				(9-14) • Poor (0-8)		
3	*Business Analytic queries design (Individual marks awarded))	30%	• Clear and proper identification of information needs • Flexible query to cater for variety of inputs, use of multiple tables • Meaningful report handlings • Data values formatted accordingly	• Excellent (25-30) • Good (16-24) • Moderate (9-15) • Poor (0-8)		3
4	Assignment Report	10%	• Comprehensive coverage • Quality of report presented • All tasks numbered, header / footer used, proper formatting	• Excellent (9-10) • Good (7-8) • Moderate (4-6) • Poor (0-3)		1

Group Member: **Task 3 marks Total marks**

1. **(Eugene Tan Yu Xian) ( )**

2. **(Joash Alwinn Voon Dirui) ( )**

3. **(Loh Jia Shou) ( )**

4. **(Ng Hong Han) ( )**

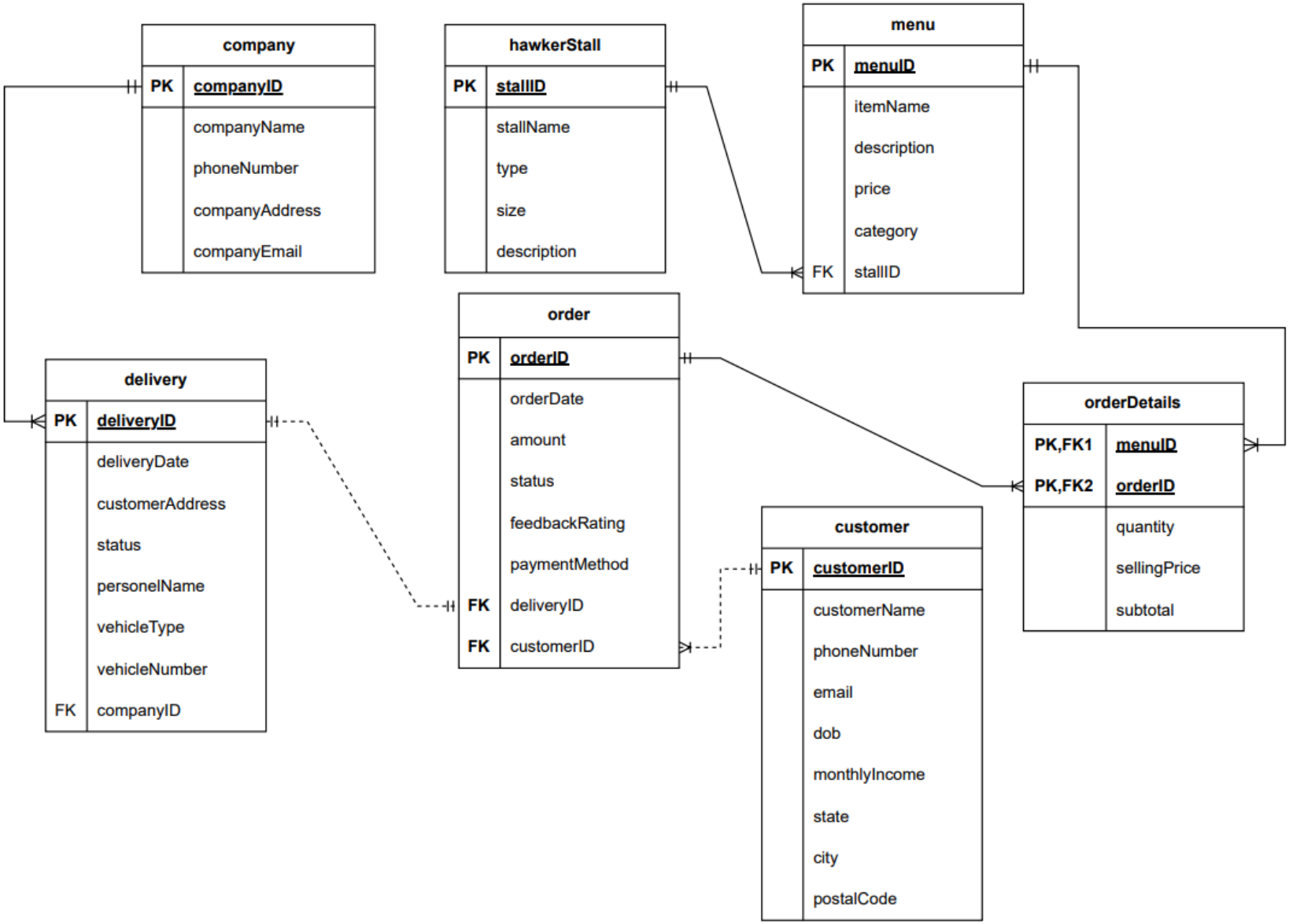
5. **(Vithiya Saraswathi a/p Scokalingam) ( )**

Table of Contents

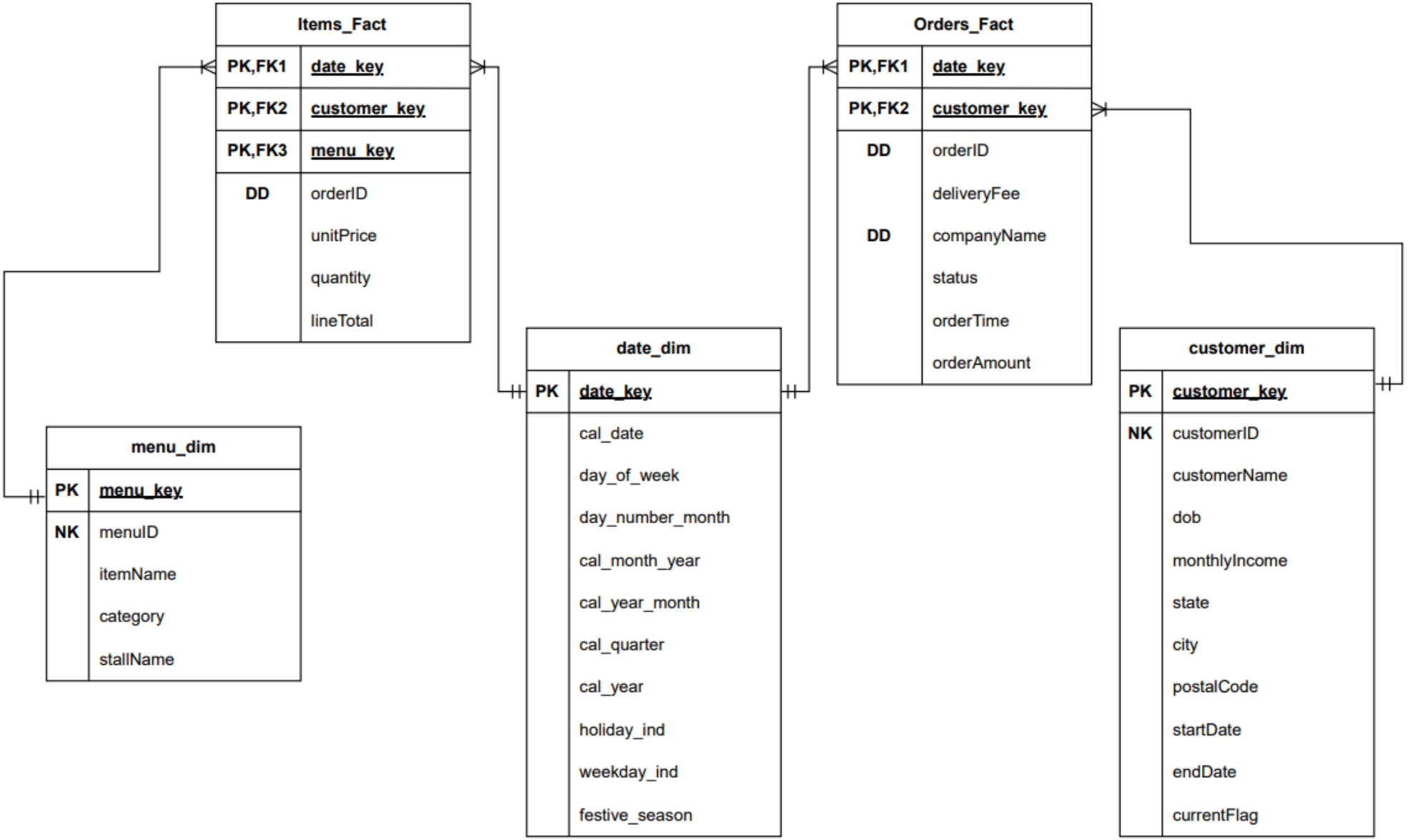
Chapter 1 Design of Data Warehouse

1.1 Logical Design

1.1.1 Original Database (Entity Relationship Diagram)



1.1.2 Star Schema Dimension and Fact Tables



1.2 Physical Design

1.2.1 Dimension Tables

Date Dimension

```
create table date_dim
(date_key              number          not null,
 cal_date              date            not null,
 day_of_week           number(1)       not null,
 day_number_month      number(2)       not null,
 cal_month_year        number(2)       not null,
 cal_year_month        char(7)         not null,
 cal_quarter           char(2)         not null,
 cal_year              number(4)       not null,
 holiday_ind           char(1)         not null,
 weekday_ind           char(1)         not null,
 festive_season        varchar(30)     ,
constraint PK_date_key primary key(date_key)
);
```

Customer Dimension

```
create table customer_dim
(customer_key   number          not null,
 customerID     number          not null,
 customerName   varchar(30)     not null,
 dob            date            not null,
 monthlyIncome  varchar(20)     not null,
 state          varchar(25)     not null,
```

```
 city           varchar(25)     not null,
 postalCode     char(5)         not null,
 startDate      date            not null,
 endDate        date            not null,
 currentFlag    number(1)       not null,
constraint PK_CustomerKey primary key(customer_key)
);
```

Menu Dimension

```
create table menu_dim
(menu_key       number     not null,
 menuID         varchar(6) not null,
 itemName  varchar(40)     not null,
 category  varchar(20)     not null,
 stallName varchar(25)     not null,
 constraint PK_menu_key primary key(menu_key)
);
```

1.2.2 Fact tables

Items_Fact

```
CREATE TABLE Items_Fact (
 date_key           number        not null,
 customer_key       number        not null,
 menu_key           number        not null,
 orderID            number        not null,
 unitPrice          number(5,2)   not null,
```

```
 quantity           number(3)     not null,
 lineTotal          number(10,2)  not null,
 constraint PK_Items_Fact_Key primary key(date_key, customer_key, menu_key, orderID),
 constraint FK_date_key foreign key (date_key) references date_dim,
 constraint FK_customer_key foreign key (customer_key) references customer_dim,
 constraint FK_menu_key foreign key (menu_key) references menu_dim
);
```

Orders_Fact

```
CREATE TABLE Orders_Fact (
 date_key           number        not null,
 customer_key       number        not null,
 orderID            number        not null,
 deliveryFee        number(4,2)   not null,
 companyName        varchar(70)   not null,
 status             varchar(10)   not null,
 orderTime          varchar(5)    not null,
 orderAmount        number(12,2)  not null,
 constraint PK_Orders_Fact_Key primary key(date_key, customer_key, orderID),
 constraint FK_date2_key foreign key (date_key) references date_dim,
 constraint FK_customer2_key foreign key (customer_key) references customer_dim
);
```

Chapter 2 Extract, Transform, Load (ETL)

2.1 Script For Initial Loading

2.1.1 Dimension Tables

Date Dimension

```
drop sequence date_seq;

create sequence date_seq
  start with 100001
  increment by 1;

set serveroutput on
declare
 startDate  date:=to_date('01/01/2015','dd/mm/yyyy');
 endDate    date:=to_date('31/12/2024','dd/mm/yyyy');
 V_DATE_KEY               NUMBER;
 V_CAL_DATE               DATE;
 V_DAY_OF_WEEK            NUMBER(1);
 V_DAY_NUM_MONTH          NUMBER(2);
 V_CAL_MONTH_YEAR         NUMBER(2);
 V_CAL_YEAR_MONTH         CHAR(7);
 V_CAL_QUARTER            CHAR(2);
 V_CAL_YEAR               NUMBER(4);
 V_HOLIDAY_IND            CHAR(1);
 V_WEEKDAY_IND            CHAR(1);
 V_FESTIVE_SEASON         VARCHAR(30);

begin
```

```
while (startDate<=endDate) loop
 V_CAL_DATE := startDate;
 V_DAY_OF_WEEK:=to_char(startDate,'D');
 V_DAY_NUM_MONTH:=to_char(startDate,'DD');
 V_CAL_MONTH_YEAR:=to_char(startDate,'MM');
 V_CAL_YEAR:=to_char(startDate,'YYYY');
 V_CAL_YEAR_MONTH:=V_CAL_YEAR||'-'||V_CAL_MONTH_YEAR;
 V_CAL_QUARTER:='Q'||to_char(startDate,'Q');
 V_HOLIDAY_IND:='N';

 if (V_DAY_OF_WEEK between 2 and 6) then
   V_WEEKDAY_IND:='Y';
 else
   V_WEEKDAY_IND:='N';
 end if;

 V_FESTIVE_SEASON := NULL;
 if startDate = to_date('01/01/' || V_CAL_YEAR, 'dd/mm/yyyy') then
    V_FESTIVE_SEASON := 'New Year';
 elsif startDate = to_date('01/05/' || V_CAL_YEAR, 'dd/mm/yyyy') then
    V_FESTIVE_SEASON := 'Labour Day';
 elsif startDate = to_date('31/08/' || V_CAL_YEAR, 'dd/mm/yyyy') then
    V_FESTIVE_SEASON := 'Merdeka Day';
 elsif startDate = to_date('25/12/' || V_CAL_YEAR, 'dd/mm/yyyy') then
    V_FESTIVE_SEASON := 'Christmas';

 -- Additional Malaysian public holidays (dates may vary annually)
 elsif startDate = to_date('29/07/2015', 'dd/mm/yyyy') then
    V_FESTIVE_SEASON := 'Hari Raya Puasa';
 elsif startDate = to_date('24/09/2015', 'dd/mm/yyyy') then
    V_FESTIVE_SEASON := 'Hari Raya Haji';
 elsif startDate = to_date('18/02/2015', 'dd/mm/yyyy') then
```

```
      V_FESTIVE_SEASON := 'Chinese New Year';
   elsif startDate = to_date('14/04/2015', 'dd/mm/yyyy') then
      V_FESTIVE_SEASON := 'Deepavali';
   end if;

   insert into date_dim values(
   date_seq.nextval,
   V_CAL_DATE,
   V_DAY_OF_WEEK,
   V_DAY_NUM_MONTH,
   V_CAL_MONTH_YEAR,
   V_CAL_YEAR_MONTH,
   V_CAL_QUARTER,
   V_CAL_YEAR,
   V_HOLIDAY_IND,
   V_WEEKDAY_IND,
   V_FESTIVE_SEASON);

   startDate:=startDate+1;
  end loop;
end;
/
```

Customer Dimension

```
drop sequence customer_dim_seq;

create sequence customer_dim_seq
start with 100001
increment by 1;

insert into customer_dim
```

```
select customer_dim_seq.nextval,
       CUSTOMERID,
       UPPER(customerName),
       dob,
        monthlyIncome,
       UPPER(state),
       UPPER(city),
       postalCode,
       '01/01/2015',
        '31/12/9999',
        1
from new_cust;
```

Menu Dimension

```
drop sequence menu_dim_seq;

create sequence menu_dim_seq
   start with 1001;

insert into menu_dim
select menu_dim_seq.nextval,
       A.menuID,
       UPPER(A.itemName),
       UPPER(A.category),
       UPPER(B.stallName)
from menu        A
join hawkerStall B on A.stallID = B.stallID;
```

2.1.2 Fact Tables

Items_Fact

```
INSERT INTO Items_Fact
SELECT
    C.date_key,
    D.customer_key,
    E.menu_key,
    A.orderID,
    B.price AS unitPrice,
    B.quantity,
    (B.price * B.quantity) AS lineTotal
FROM new_orders          A
JOIN new_order_details   B ON A.orderID = B.orderID
JOIN date_dim            C ON TRUNC(A.orderDate) = TRUNC(C.cal_date)
JOIN customer_dim        D ON A.customerID = D.customerID
JOIN menu_dim            E ON B.menuID = E.menuID
JOIN delivery            F ON A.deliveryID = F.deliveryID
JOIN company             G ON F.companyID = G.companyID;
```

Orders_Fact

```
INSERT INTO Orders_Fact
SELECT
    B.date_key,
    C.customer_key,
    A.orderID,
    A.deliveryFee,
    UPPER(E.companyName),
    UPPER(A.status),
    TO_CHAR(A.orderDate, 'HH24:MI') AS orderTime,
```

```
    0.00
FROM new_orders          A
JOIN date_dim            B ON TRUNC(A.orderDate) = TRUNC(B.cal_date)
JOIN customer_dim        C ON A.customerID = C.customerID
JOIN delivery            D ON A.deliveryID = D.deliveryID
JOIN company             E ON D.companyID = E.companyID;

-- this table is created to minimise the loadwork when calculating the sum from items_fact
-- straight to updating the orders_fact table. Instead we insert into a temp_table first then
-- only update

create table temp_table(
orderID number           not null,
orderAmount number(12,2) not null
);

insert into temp_table
select orderID, SUM(lineTotal) orderAmount
from items_fact
group by orderID;

update orders_fact A
set orderAmount = (select orderAmount
                   from temp_table
                   where orderID = A.orderID);
```

2.2 Script For Subsequent Loading

2.2.1 Dimension Tables

Insert New Customer

```
drop procedure prod_new_customer;
create or replace procedure prod_new_customer is
begin
    insert into customer_dim
    select customer_dim_seq.nextval,
           customerID,
           UPPER(customerName),
           dob,
           monthlyIncome,
        UPPER(state),
        UPPER(city),
        postalCode,
           '01/01/2015',
           '31/12/9999',
           1
     from new_cust
     where customerID not in (
         select customerID from customer_dim
     );
end;
/

EXEC prod_new_customer
```

Insert New Menu Item

```
drop procedure prod_new_menu_item;
create or replace procedure prod_new_menu_item is
begin
    insert into menu_dim
    select menu_dim_seq.nextval,
           A.menuID,
           UPPER(A.itemName),
           UPPER(A.category),
           UPPER(B.stallName)
     from menu         A
     join hawkerStall  B on A.stallID = B.stallID
     where menuID not in (
         select menuID from menu_dim
     );
end;
/

EXEC prod_new_menu_item
```

2.2.2 Fact Tables

Insert New Item_Facts data

```
drop procedure prod_new_items_fact;

create or replace procedure prod_new_items_fact is
begin
    insert into Items_Fact
    select C.date_key,
           D.customer_key,
```

```
            E.menu_key,
            A.orderID,
            B.price AS unitPrice,
            B.quantity,
            (B.price * B.quantity) AS lineTotal
    from new_orders         A
    join new_order_details  B ON A.orderID = B.orderID
    join date_dim           C ON TRUNC(A.orderDate) = TRUNC(C.cal_date)
    join customer_dim       D ON A.customerID = D.customerID
    join menu_dim           E ON B.menuID = E.menuID
    left join Items_Fact IF ON IF.date_key = C.date_key
    and IF.customer_key = D.customer_key
    and IF.menu_key = E.menu_key
    and IF.orderID = A.orderID
    where IF.date_key is null;
end;
/

EXEC prod_new_items_fact
```

Insert New Orders_Fact data

```
drop procedure prod_load_temp_table;
drop procedure prod_load_orders_fact;

create or replace procedure prod_load_temp_table is
begin
    delete temp_table;

    insert into temp_table (orderID, orderAmount)
    select orderID,
           SUM(lineTotal) AS orderAmount
    from items_fact
```

```
    group by orderID;
end;
/

EXEC prod_load_temp_table

create or replace procedure prod_load_orders_fact is
begin
    insert into Orders_Fact
    select B.date_key,
           C.customer_key,
            A.orderID,
            A.deliveryFee,
            UPPER(E.companyName),
            UPPER(A.status),
            TO_CHAR(A.orderDate, 'HH24:MI') AS orderTime,
            0.00
    from new_orders     A
    join date_dim       B ON TRUNC(A.orderDate) = TRUNC(B.cal_date)
    join customer_dim   C ON A.customerID = C.customerID
    join delivery       D ON A.deliveryID = D.deliveryID
    join company        E ON D.companyID = E.companyID
    where not exists (
        select 1
        from Orders_Fact F
        where F.date_key = B.date_key
        and F.customer_key = C.customer_key
        and F.orderID = A.orderID
    );

    update Orders_Fact G
    set orderAmount = (select orderAmount
```

```
                        from temp_table H
                        where G.orderID = H.orderID);
end;
/

EXEC prod_load_orders_fact
```

2.3 Type 2 Slow Changing Dimension (SCD) Maintenance

2.3.1 Update Customer's endDate and currentFlag

```
UPDATE customer_dim
SET endDate = SYSDATE, currentFlag = 0
WHERE customerID = 112800
AND currentFlag = 1;
```

2.3.2 Insert New Row Based On Desired Changes

```
INSERT INTO customer_dim (customer_key, customerID, customerName, dob, monthlyIncome, state, city, postalCode,
startDate, endDate, currentFlag)
VALUES (customer_dim_seq.nextval, 112800, 'LORENA PURCHALL', TO_DATE('14/01/1996', 'dd/mm/yyyy'), '7001 - 9000',
'KELANTAN', 'KOTA BHARU', '15550', SYSDATE, TO_DATE('31/12/9999', 'dd/mm/yyyy'), 1);
```

Chapter 3 Business Analytics Reports

3.1 Eugene Tan Yu Xian

3.1.1 Sales Analysis for Holidays vs. Non-Holidays (2023 vs. 2024)

SQL:

```
SPOOL 'C:\Users\Asus\Downloads\dw\Q1_output.txt'

-- Step 1: Set up the title, page number, and date formatting
SET SERVEROUTPUT ON
SET PAGESIZE 50;
SET LINESIZE 120;

ALTER SESSION SET NLS_DATE_FORMAT = 'dd/MM/YYYY';

-- Using TTITLE to display the title, date, and page number
TTITLE -
CENTER 'Business Analytics Report - Sales Analysis for Holidays vs. Non-Holidays (2023 vs. 2024)' -
skip 1 -
RIGHT 'Date: ' _DATE -
SKIP 1 -
RIGHT 'Page ' FORMAT 999 SQL.PNO -
SKIP 2;

-- Step 2: Set column formatting for readability
COLUMN period_type FORMAT A15 HEADING "Period Type";
COLUMN holiday_name FORMAT A17 HEADING "Holiday Name";
COLUMN total_orders_2023 FORMAT 999,999 HEADING "Total|Orders (2023)";
COLUMN total_orders_2024 FORMAT 999,999 HEADING "Total|Orders (2024)";
COLUMN total_sales_2023 FORMAT 999,999,999.99 HEADING "Total Sales|(RM) 2023";
COLUMN total_sales_2024 FORMAT 999,999,999.99 HEADING "Total Sales|(RM) 2024";
```

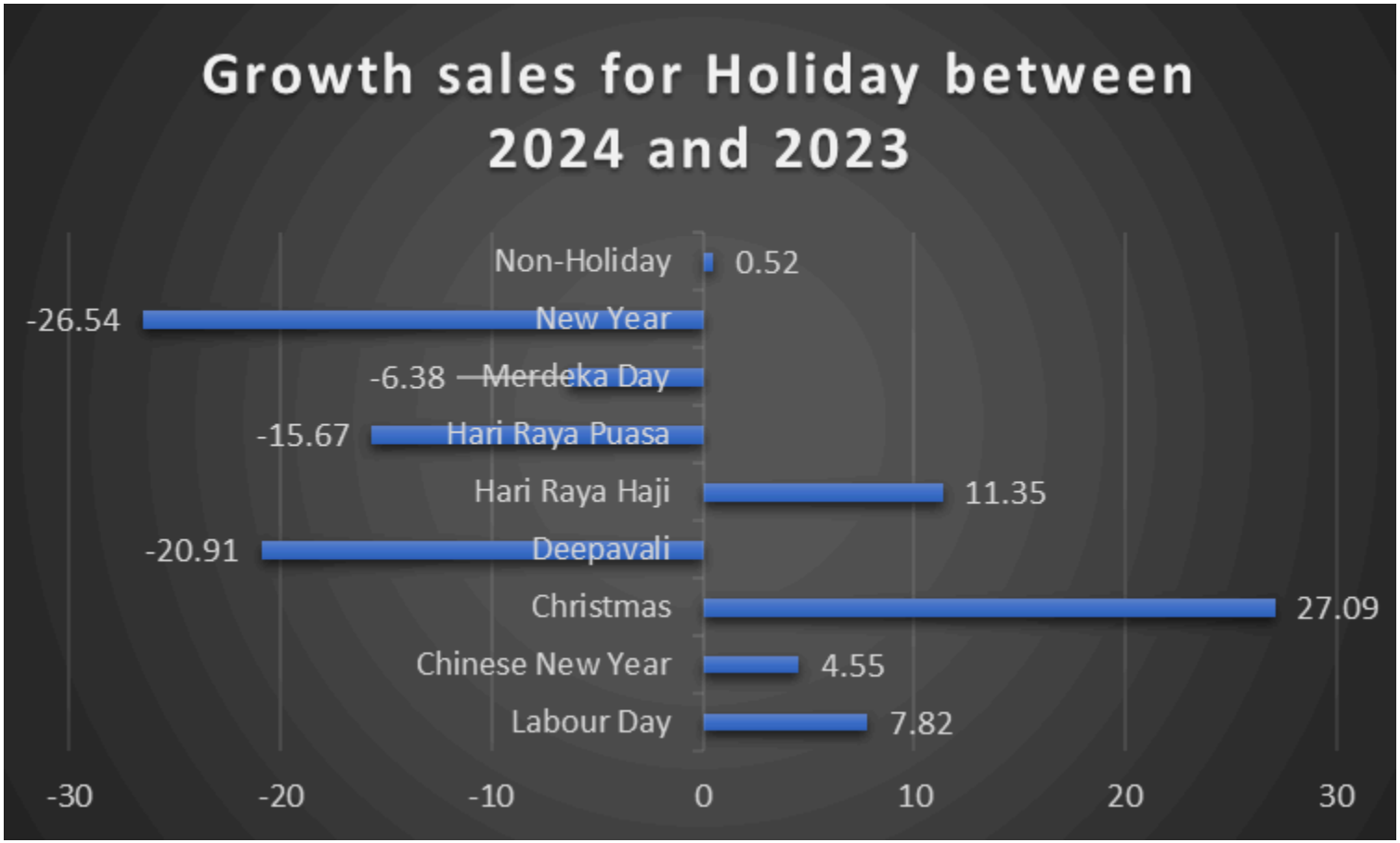
```
COLUMN sales_percentage FORMAT 999.99 HEADING "Sales|Percentage(%)";
COLUMN sales_growth FORMAT 999.99 HEADING "Sales|Growth(%)";
COLUMN order_growth FORMAT 999.99 HEADING "Order|Growth(%)";


-- Step 3: Break on Year and Compute Totals
-- BREAK ON year SKIP 1;
COMPUTE SUM LABEL 'Total' OF total_orders_2023 total_orders_2024 on sales_growth;

-- Step 4: Generate the sales analysis for holidays vs. non-holidays for the years 2023 and 2024
WITH sales_analysis AS (
    SELECT
        CASE
            -- Manually set 'Holiday' for known festive seasons
            WHEN dd.festive_season IN ('New Year', 'Labour Day', 'Merdeka Day', 'Christmas',
                                       'Hari Raya Puasa', 'Hari Raya Haji', 'Chinese New Year',
'Deepavali')
            THEN 'Y'
            ELSE 'N'
        END AS holiday_flag,   -- Override holiday indicator based on festive season
        COALESCE(dd.festive_season, 'Non-Holiday') AS holiday_name, -- Festive season name, or
'Non-Holiday'
        COUNT(ofact.orderID) AS total_orders, -- Total number of orders during this period
        SUM(ofact.orderAmount) AS total_sales, -- Total sales amount during this period
        AVG(ofact.orderAmount) AS avg_sales_per_order, -- Average sales per order
        dd.cal_year AS year, -- Use CAL_YEAR from date dimension
        SUM(SUM(ofact.orderAmount)) OVER (PARTITION BY dd.cal_year) AS overall_sales -- Calculate
total sales for each year
    FROM
        Orders_Fact ofact
    JOIN
        date_dim dd
    ON
        ofact.date_key = dd.date_key
```

```
    WHERE
        dd.cal_year BETWEEN 2023 AND 2024
    GROUP BY
        dd.festive_season, dd.cal_year
),
-- Step 5: Add ranking based on total sales and calculate sales percentage for each year
ranked_sales AS (
    SELECT
        CASE
            WHEN holiday_flag = 'Y' THEN 'Holiday'
            ELSE 'Non-Holiday'
        END AS period_type,  -- Differentiating holiday and non-holiday periods
        holiday_name, -- Holiday name (or 'Non-Holiday')
        total_orders,  -- Total number of orders
        total_sales,  -- Total sales value
        total_sales / overall_sales * 100 AS sales_percentage, -- Calculate percentage of total sales
without rounding first
        ROUND(avg_sales_per_order, 2) AS avg_sales_per_order, -- Average order amount
        year,  -- Include the year in the results
        RANK() OVER (PARTITION BY year ORDER BY total_sales DESC) AS rank -- Ranking based on total
sales for each year
    FROM
        sales_analysis
),
-- Step 6: Calculate percentage growth for sales and orders between 2023 and 2024
growth_comparison AS (
    SELECT
        r1.period_type,
        r1.holiday_name,
        r1.total_orders AS total_orders_2023,
        r2.total_orders AS total_orders_2024,
        r1.total_sales AS total_sales_2023,
        r2.total_sales AS total_sales_2024,
        -- Calculate growth percentage for sales
```

```
        CASE WHEN r1.total_sales > 0
             THEN ROUND(((r2.total_sales - r1.total_sales) / r1.total_sales) * 100, 2)
             ELSE NULL END AS sales_growth,
        -- Calculate growth percentage for orders
        CASE WHEN r1.total_orders > 0
             THEN ROUND(((r2.total_orders - r1.total_orders) / r1.total_orders) * 100, 2)
             ELSE NULL END AS order_growth
    FROM ranked_sales r1
    JOIN ranked_sales r2
    ON r1.period_type = r2.period_type AND r1.holiday_name = r2.holiday_name
    WHERE r1.year = 2023 AND r2.year = 2024
)
-- Step 7: Output formatted report with ranking, sales percentage, and year
SELECT
    g.period_type, -- Holiday or Non-Holiday
    g.holiday_name, -- Holiday name
    g.total_orders_2023,
    g.total_orders_2024,
    g.total_sales_2023,
    g.total_sales_2024,
    g.sales_growth,
    g.order_growth
FROM
    growth_comparison g
ORDER BY
    g.sales_growth DESC;

-- Step 8: Clear the title after the report
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF
SPOOL OFF;
```

OUTPUT:

```
Session altered.

                Business Analytics Report - Sales Analysis for Holidays vs. Non-Holidays (2023 vs. 2024)
                                                                                Date: 24/09/2024
                                                                                       Page    1
                                    Total         Total  Total Sales  Total Sales  Sales    Order
Period Type  Holiday Name      Orders (2023) Orders (2024)    (RM) 2023    (RM) 2024  Growth(%)Growth(%)
------------ ----------------  ------------- ------------- ------------ ------------ --------- --------
Holiday      Christmas                  53            59       14,527.50    18,463.50   27.09     11.32
Holiday      Hari Raya Haji             52            53       14,412.50    16,048.00   11.35      1.92
Holiday      Labour Day                 53            54       13,832.00    14,914.00    7.82      1.89
Holiday      Chinese New Year           52            56       14,236.00    14,884.00    4.55      7.69
Non-Holiday  Non-Holiday            19,646        19,728    5,532,102.00 5,560,784.50     .52       .42
Holiday      Merdeka Day                53            55       15,073.50    14,111.50   -6.38      3.77
Holiday      Hari Raya Puasa            59            51       16,848.00    14,208.00  -15.67    -13.56
Holiday      Deepavali                  58            52       16,283.50    12,879.00  -20.91    -10.34
Holiday      New Year                   55            51       16,708.50    12,274.50  -26.54     -7.27

9 rows selected.
```


Growth sales for Holiday between 2024 and 2023
Non-Holiday
0.52
-26.54
New Year
-6.38
Merdeka Day
-15.67
Hari Raya Puasa
Hari Raya Haji
11.35
-20.91
Deepavali
Christmas
27.09
Chinese New Year
4.55
Labour Day
7.82
-30
-20
-10
0
10
20
30

This bar chart illustrates the percentage growth in sales for various holidays between 2023 and 2024. Holidays like Christmas (+27.09%), Hari Raya Haji (+11.35%), Labour Day (+7.82%), and Chinese New Year (+4.55%) saw positive growth, with Christmas experiencing the highest increase. On the other hand, significant declines were observed during Deepavali (-20.91%), New Year (-26.54%), and Hari Raya Puasa (-15.67%). Non-holiday periods showed only slight growth (+0.52%). The chart highlights which holiday seasons performed better or worse in terms of sales compared to the previous year.

Action or recommendation can be made if we can take more attention on holidays that have declined sales. Offering combo meals or packages at a slight discount can encourage customers to buy more, helping to increase overall sales.

3.1.2 Top 10% Order Value Analysis with Delivery Fee Between 2023 AND 2024 with Predictions

SQL:

```
SPOOL 'C:\Users\Asus\Downloads\dw\Q2_output.txt'

-- Step 1: Set up title, page formatting, and date
SET PAGESIZE 50;
SET LINESIZE 130;
ALTER SESSION SET NLS_DATE_FORMAT = 'dd/MM/YYYY';

-- Using TTITLE to display title, date, and page number
TTITLE -
CENTER 'Business Analytics Report - Top 10% Order Value Analysis with Delivery Fee Between 2023 AND
2024 with Predictions' -
SKIP 1 -
RIGHT 'Date: ' _DATE -
SKIP 1 -
RIGHT 'Page ' FORMAT 999 SQL.PNO -
SKIP 2

-- Set column formatting
```

```
COLUMN fee_range FORMAT A21 HEADING "Delivery Fee Range";
COLUMN total_orders_2023 FORMAT 999,999 HEADING "Top 10%|Orders 2023";
COLUMN total_orders_2024 FORMAT 999,999 HEADING "Top 10%|Orders 2024";
COLUMN total_revenue_2023 FORMAT 999,999,999.99 HEADING "Total Revenue|2023 (RM)";
COLUMN total_revenue_2024 FORMAT 999,999,999.99 HEADING "Total Revenue|2024 (RM)";
COLUMN revenue_growth FORMAT 999.99 HEADING "Revenue|Growth (%)";
COLUMN predicted_revenue_2025 FORMAT 999,999,999.99 HEADING "Predicted|Revenue 2025|(RM)";
COLUMN action FORMAT A18 HEADING "Action";

BREAK ON fee_range SKIP 1;

-- Step 2: Identify large orders (top 10% based on order value)
WITH percentile_order AS (
    SELECT PERCENTILE_CONT(0.90) WITHIN GROUP (ORDER BY ofact.orderAmount) AS large_order_threshold
    FROM Orders_Fact ofact
    JOIN date_dim dd ON ofact.date_key = dd.date_key
    WHERE dd.cal_year BETWEEN 2023 AND 2024
),
-- Step 3: Calculate total orders and revenue for 2023 and 2024 for large orders, including delivery
fee
order_comparison AS (
    SELECT
        CASE
            WHEN ofact.deliveryFee <= 2 THEN 'Low Fee (<= 2 RM)'
            WHEN ofact.deliveryFee BETWEEN 2 AND 5 THEN 'Medium Fee (2 - 5 RM)'
            ELSE 'High Fee (> 5 RM)'
        END AS fee_range,
        SUM(CASE WHEN dd.cal_year = 2023 THEN 1 ELSE 0 END) AS total_orders_2023,
        SUM(CASE WHEN dd.cal_year = 2024 THEN 1 ELSE 0 END) AS total_orders_2024,
        SUM(CASE WHEN dd.cal_year = 2023 THEN ofact.orderAmount + ofact.deliveryFee ELSE 0 END) AS
total_revenue_2023,
        SUM(CASE WHEN dd.cal_year = 2024 THEN ofact.orderAmount + ofact.deliveryFee ELSE 0 END) AS
total_revenue_2024,
        CASE
```

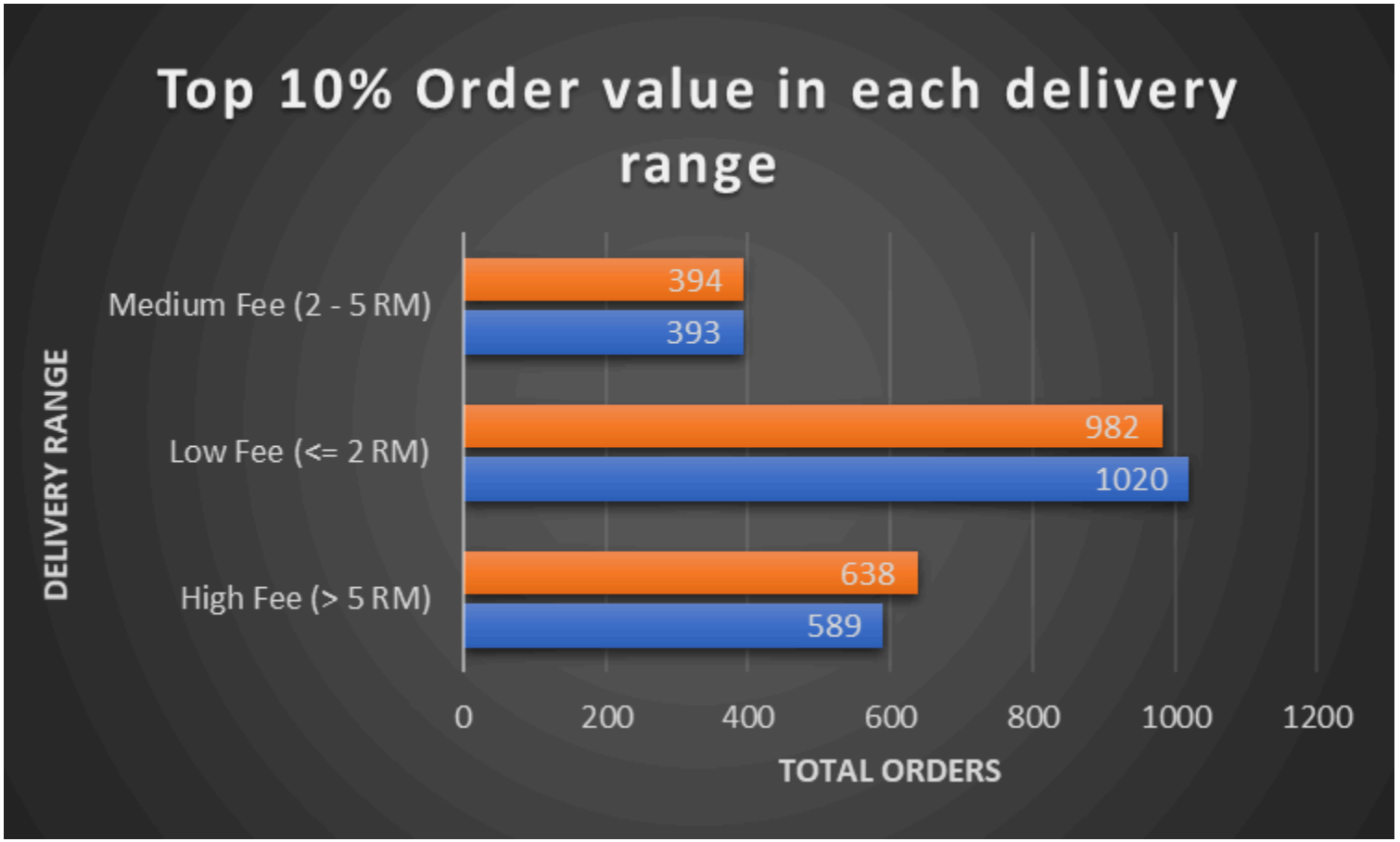
```
            WHEN SUM(CASE WHEN dd.cal_year = 2023 THEN ofact.orderAmount + ofact.deliveryFee ELSE 0
END) > 0
            THEN ((SUM(CASE WHEN dd.cal_year = 2024 THEN ofact.orderAmount + ofact.deliveryFee ELSE 0
END) -
                   SUM(CASE WHEN dd.cal_year = 2023 THEN ofact.orderAmount + ofact.deliveryFee ELSE 0
END)) /
                   SUM(CASE WHEN dd.cal_year = 2023 THEN ofact.orderAmount + ofact.deliveryFee ELSE 0
END)) * 100
            ELSE 0
        END AS revenue_growth
    FROM Orders_Fact ofact
    JOIN date_dim dd ON ofact.date_key = dd.date_key
    WHERE ofact.orderAmount > (SELECT large_order_threshold FROM percentile_order)
    GROUP BY
        CASE
            WHEN ofact.deliveryFee <= 2 THEN 'Low Fee (<= 2 RM)'
            WHEN ofact.deliveryFee BETWEEN 2 AND 5 THEN 'Medium Fee (2 - 5 RM)'
            ELSE 'High Fee (> 5 RM)'
        END
)
-- Step 4: Add action recommendation based on revenue growth
SELECT
    fee_range,
    total_orders_2023,
    total_orders_2024,
    ROUND(total_revenue_2023, 2) AS total_revenue_2023,
    ROUND(total_revenue_2024, 2) AS total_revenue_2024,
    ROUND(revenue_growth, 2) AS revenue_growth,
    CASE
        WHEN revenue_growth > 0 THEN ROUND(total_revenue_2024 * (1 + (revenue_growth / 100)), 2)
        ELSE ROUND(total_revenue_2024 * (1 + (revenue_growth / 100)), 2) -- Apply a 5% reduction for
negative growth prediction
    END AS predicted_revenue_2025,
    CASE
```

```
        WHEN revenue_growth > 0 THEN 'Free Delivery Fee'
        ELSE 'Lower Delivery Fee'
    END AS action
FROM order_comparison
ORDER BY fee_range;

-- Clear formatting
TTITLE OFF;
SPOOL OFF;
```

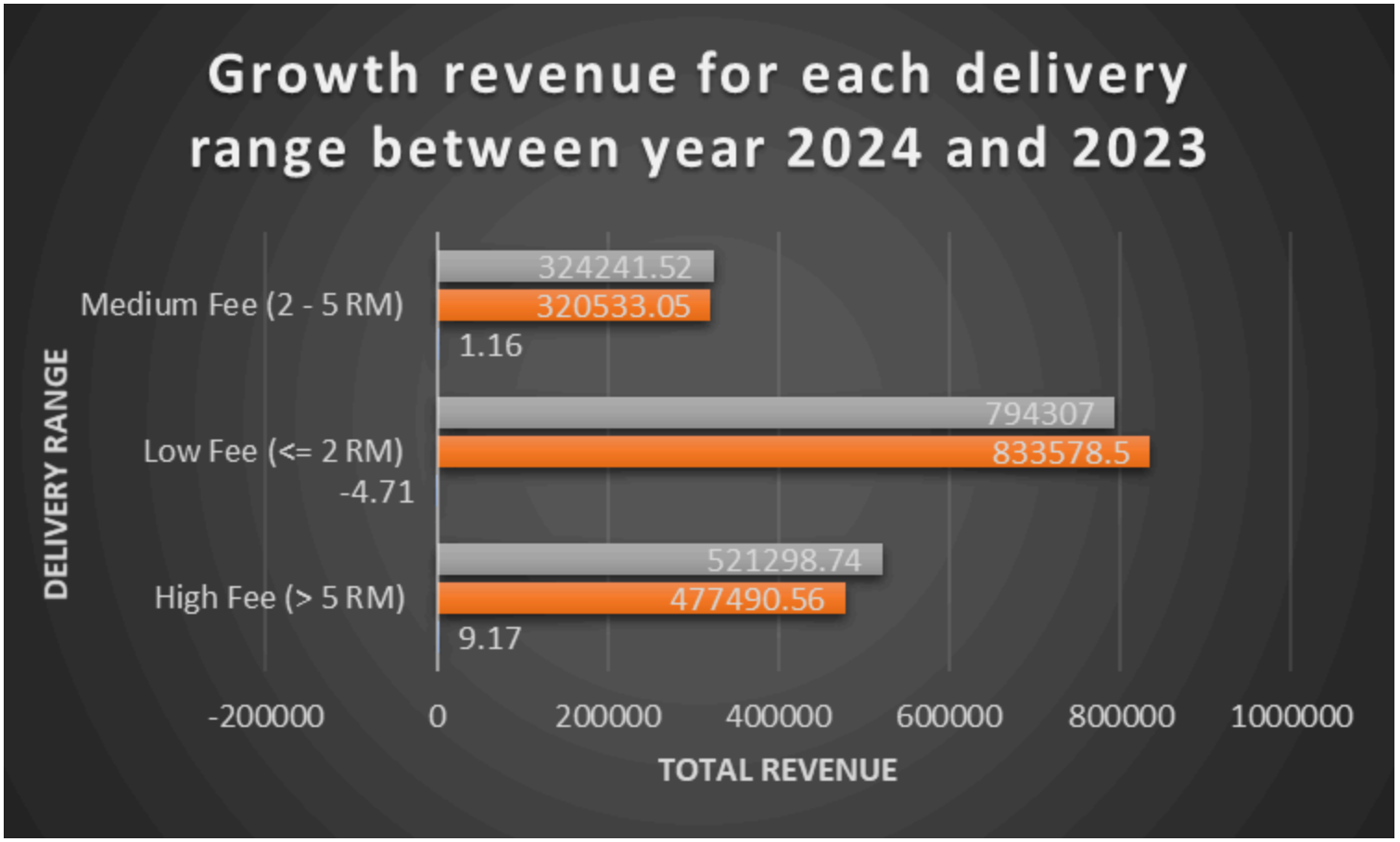
OUTPUT:

```
         Business Analytics Report - Delivery Fee and Top 10% Order Value Analysis Between 2023 AND 2024 with
Predictions

Date:26/09/2024

Page    1
```

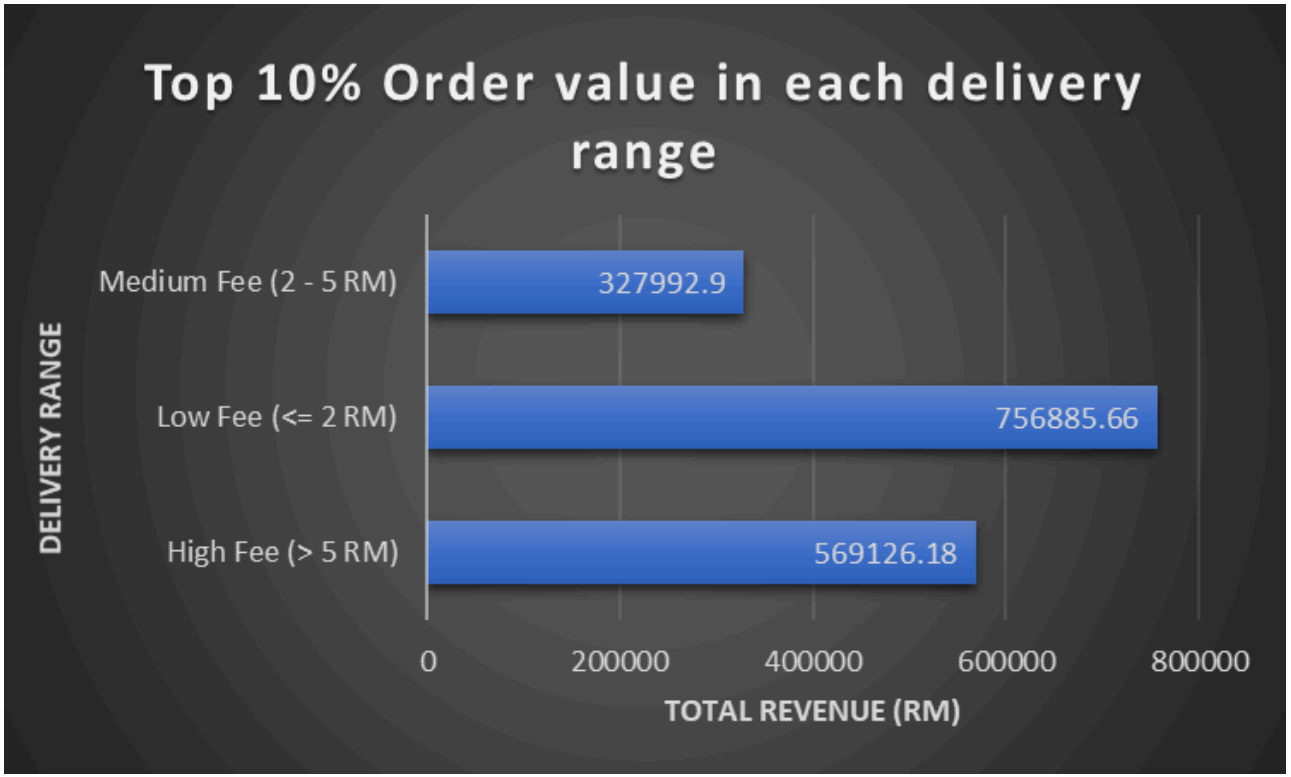
Delivery Fee Range	Top 10% Orders 2023	Top 10% Orders 2024	Total Revenue 2023 (RM)	Total Revenue 2024 (RM)	Revenue Growth (%)	Predicted Revenue 2025 (RM)	Action
High Fee (> 5 RM)	589	638	477,490.56	521,298.74	9.17	569,126.18	Free Delivery Fee
Low Fee (<= 2 RM)	1,020	982	833,578.50	794,307.00	-4.71	754,591.65	Lower DeliveryFee
Medium Fee (2 - 5 RM)	393	394	320,533.05	324,241.52	1.16	327,992.90	Free Delivery Fee



So we this report was made to provide benefit to the top 10 % of our order value for each delivery range.



This bar chart shows the growth in revenue for different delivery fee ranges between 2023 and 2024. The chart compares revenue for three fee ranges: Medium Fee (2–5 RM), Low Fee (≤ 2 RM), and High Fee (> 5 RM). High Fee deliveries saw the highest growth rate of 9.17%, while Medium Fee deliveries had modest growth at 1.16%. However, Low Fee deliveries experienced a decline with a negative growth rate of -4.71%. The revenue numbers are displayed for both years, with Low Fee deliveries showing the largest total revenue in both years, despite the negative growth.



This bar graph shows the predicted revenue for 2025. We analyse the predicted revenue of 2025 by using the growth revenue from 2023 to 2024. Based on the value shown in the graph we make an action which provides a free delivery fee for the top 10% of the order value in 2025 . We estimated that 2025 will make more revenue than 2024 which will also make profit while we provide free delivery fee. For the decline revenue for low fee is stated that it is the most revenue made for our business. What we should do is keep track of sales while providing feedback for reviewing the customer satisfaction of our delivery fee , time , distance, etc.

3.1.3 Income Analysis For Each State

SQL:

```
SPOOL 'C:\Users\Asus\Downloads\dw\Q3_output.txt'

-- Step 1: Set up the title, page formatting, and date
SET PAGESIZE 50;
SET LINESIZE 110;
ALTER SESSION SET NLS_DATE_FORMAT = 'dd/MM/YYYY';

-- Using TTITLE to display the title, date, and page number
TTITLE -
CENTER 'Business Analytics Report - Income Analysis For Each State' -
skip 1 -
RIGHT 'Date: ' _DATE -
SKIP 1 -
RIGHT 'Page ' FORMAT 999 SQL.PNO -
SKIP 2

-- Set column formatting for readability
COLUMN state_rank FORMAT 999 HEADING  "State Rank";
COLUMN income_group_rank FORMAT 999 HEADING "Income | Group Rank";
COLUMN state FORMAT A15 HEADING "State";
COLUMN income_group FORMAT A20 HEADING "Income | Group";
COLUMN total_orders  FORMAT 999,999 HEADING "Total | Orders";
COLUMN total_sales FORMAT 999,999,999.99 HEADING "Total | Sales (RM)";
COLUMN sales_percentage FORMAT 999.99 HEADING "Sales | Percentage (%)" ;

WITH state_total_sales AS (
    -- Calculate total sales per state and overall grand total sales
```

```
    SELECT
        cd.state,
        COUNT(ofact.orderID) AS total_orders,
        SUM(ofact.orderAmount) AS total_sales,
        SUM(SUM(ofact.orderAmount)) OVER () AS grand_total_sales -- Grand total for all states
    FROM Orders_Fact ofact
    JOIN customer_dim cd ON ofact.customer_key = cd.customer_key
    GROUP BY cd.state
),
ranked_states AS (
    -- Rank states by total sales
    SELECT
        state,
        total_orders,
        total_sales,
        (total_sales / grand_total_sales) * 100 AS state_sales_percentage,
        RANK() OVER (ORDER BY total_sales DESC) AS state_rank -- Rank states by total sales
    FROM state_total_sales
),
customer_spending AS (
    -- Use monthlyIncome as a string (range) and calculate total orders and total sales per income
range within each state
    SELECT
        cd.state,
        cd.monthlyIncome AS income_group,  -- Use monthlyIncome directly as the income group
        COUNT(ofact.orderID) AS total_orders,
        SUM(ofact.orderAmount) AS total_sales,
        SUM(SUM(ofact.orderAmount)) OVER (PARTITION BY cd.state) AS state_total_sales -- Total sales
per state
    FROM Orders_Fact ofact
    JOIN customer_dim cd ON ofact.customer_key = cd.customer_key
    GROUP BY cd.state, cd.monthlyIncome  -- Group by state and monthly income range
),
ranked_income_groups AS (
```

```
    -- Rank income groups within each state by total sales
    SELECT
        state,
        income_group,
        total_orders,
        total_sales,
        (total_sales / state_total_sales) * 100 AS income_group_sales_percentage,
        RANK() OVER (PARTITION BY state ORDER BY total_sales DESC) AS income_group_rank
    FROM customer_spending
),
row_numbered AS (
    -- Add a row number within each state to only show the state and rank once
    SELECT
        rs.state_rank,
        rig.income_group_rank,
        rs.state,
        rig.income_group,
        rig.total_orders,
        ROUND(rig.total_sales, 2) AS total_sales,
        ROUND(rig.income_group_sales_percentage, 2) AS sales_percentage,
        ROW_NUMBER() OVER (PARTITION BY rig.state ORDER BY rig.income_group_rank) AS row_num
    FROM ranked_income_groups rig
    JOIN ranked_states rs ON rig.state = rs.state
)
-- Step 3: Output the formatted report with conditional state and rank display
SELECT
    CASE WHEN row_num = 1 THEN state_rank ELSE NULL END AS "State Rank",
    CASE WHEN row_num = 1 THEN state ELSE NULL END AS "State",
    income_group_rank,
    income_group,
    total_orders,
    total_sales,
    sales_percentage
FROM row_numbered
```

```
ORDER BY state_rank, income_group_rank;

-- Clear any formatting changes after the report
TTITLE OFF;
SPOOL OFF;
```

OUTPUT:

```
Session altered.
```

Business Analytics Report - Income Analysis For Each State

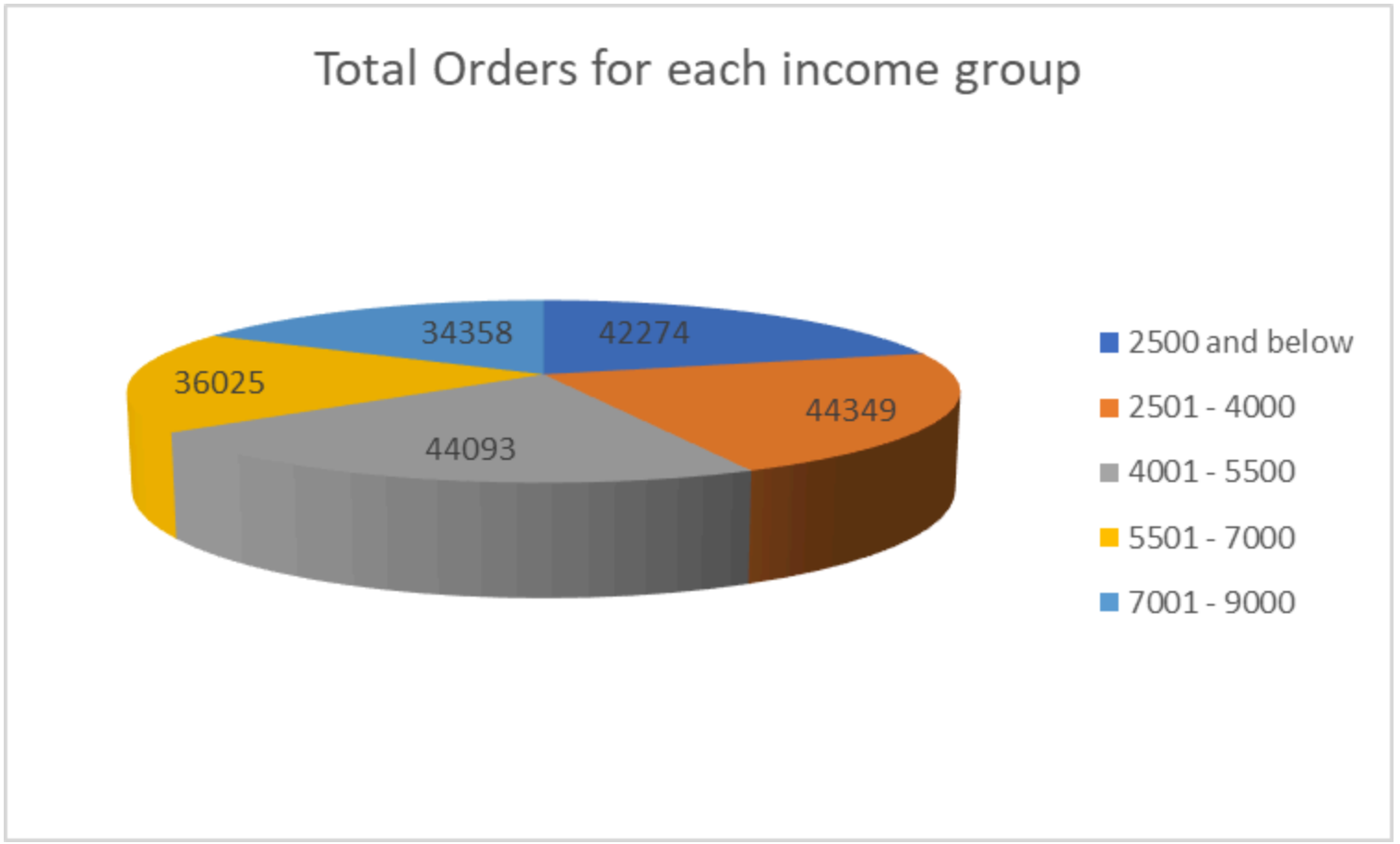
Date: 21/09/2024
Page 1

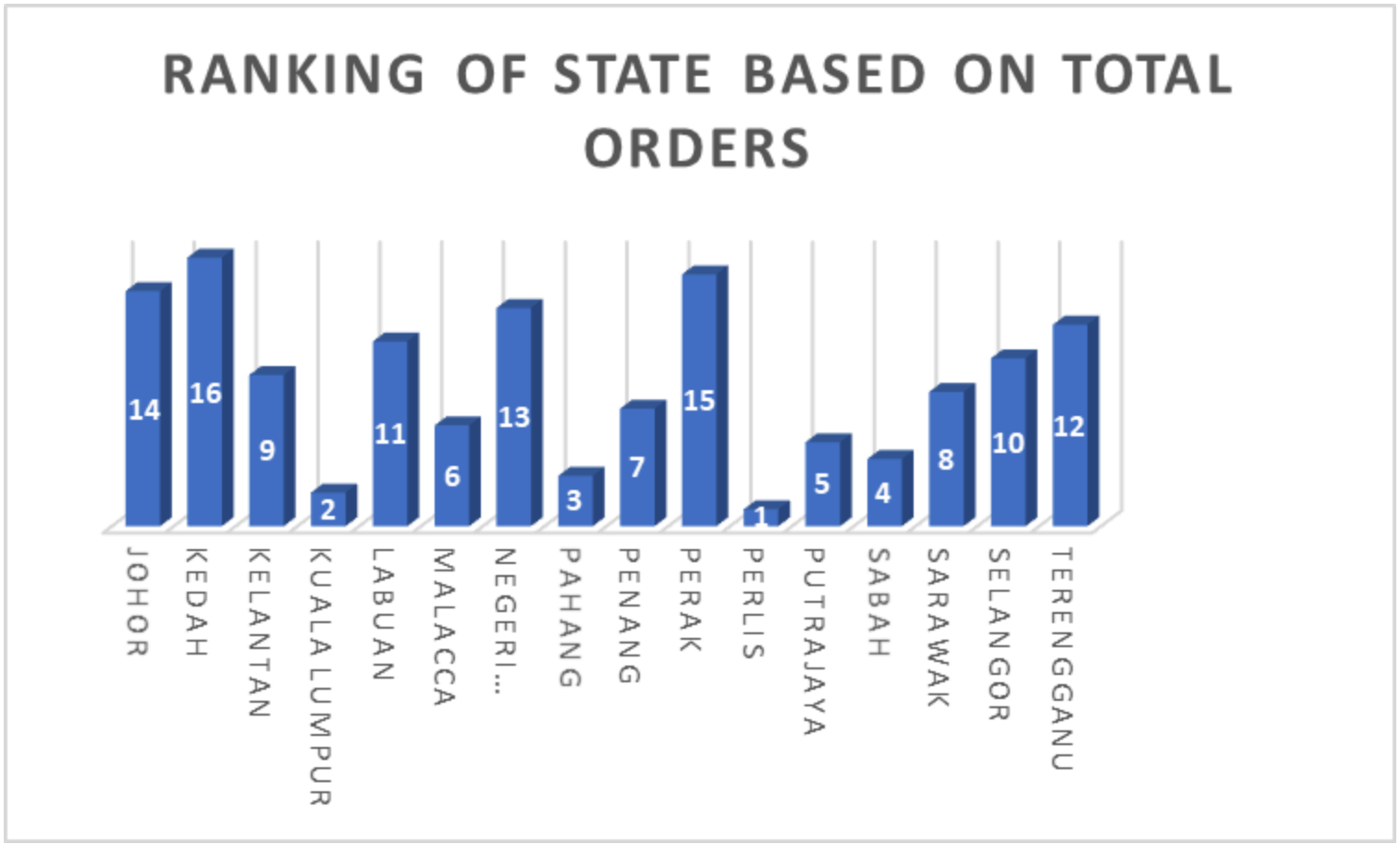
State Rank	State	Income Group Rank	Income Group	Total Orders	Total Sales (RM)	Sales Percentage (%)
1	PERLIS	1	5501 - 7000	6,118	1,729,779.00	33.68
		2	7001 - 9000	4,059	1,146,336.50	22.32
		3	4001 - 5500	4,013	1,128,835.00	21.98
		4	2500 and below	2,045	568,972.00	11.08
		5	2501 - 4000	2,007	561,584.50	10.94
2	KUALA LUMPUR	1	4001 - 5500	10,106	2,853,116.00	61.68
		2	2501 - 4000	2,103	618,043.00	13.36
		3	7001 - 9000	1,967	579,230.50	12.52
		4	2500 and below	2,013	575,443.50	12.44
3	PAHANG	1	4001 - 5500	6,004	1,677,532.00	36.55
		2	5501 - 7000	4,134	1,157,411.00	25.22
		3	2501 - 4000	2,049	606,121.50	13.21

	4	2500 and below	2,072	587,038.00	12.79
	5	7001 - 9000	2,019	561,213.50	12.23
4 SABAH	1	2501 - 4000	7,896	2,247,691.50	49.52
	2	7001 - 9000	2,062	600,711.00	13.24
	3	2500 and below	2,042	576,889.00	12.71
	4	4001 - 5500	2,049	565,853.50	12.47
	5	5501 - 7000	1,932	547,458.00	12.06
5 PUTRAJAYA	1	2500 and below	4,011	1,165,015.00	28.93
	2	7001 - 9000	3,975	1,124,368.50	27.92
	3	2501 - 4000	2,065	598,116.50	14.85
	4	4001 - 5500	2,047	576,582.50	14.32
	5	5501 - 7000	1,983	562,869.50	13.98
6 MALACCA	1	4001 - 5500	3,972	1,135,122.50	28.27
	2	5501 - 7000	3,982	1,132,582.00	28.20
	3	2501 - 4000	2,108	602,508.50	15.00
	4	7001 - 9000	2,035	592,284.00	14.75
	5	2500 and below	1,978	553,104.00	13.77
7 PENANG	1	4001 - 5500	5,965	1,703,378.00	42.64
	2	7001 - 9000	4,009	1,166,775.00	29.21
	3	5501 - 7000	1,983	565,969.00	14.17
	4	2501 - 4000	2,037	558,744.00	13.99
8 SARAWAK	1	7001 - 9000	4,194	1,164,486.00	34.03
	2	2500 and below	3,993	1,133,870.00	33.14
	3	4001 - 5500	1,978	568,275.50	16.61
	4	5501 - 7000	1,990	555,303.00	16.23
9 KELANTAN	1	5501 - 7000	5,896	1,676,354.50	50.33
	2	2501 - 4000	2,043	569,359.00	17.10
	3	4001 - 5500	1,942	545,719.00	16.39
	4	2500 and below	1,918	539,087.50	16.19
10 SELANGOR	1	2500 and below	4,111	1,169,224.00	40.77

State Rank	State	Income Group Rank	Income Group	Total Orders	Total Sales (RM)	Sales Percentage (%)
		2	7001 - 9000	2,040	587,322.00	20.48
		3	2501 - 4000	2,032	564,353.00	19.68
		4	5501 - 7000	1,959	546,893.50	19.07
11	LABUAN	1	2500 and below	4,059	1,158,382.00	40.48
		2	7001 - 9000	4,003	1,132,281.50	39.57
		3	2501 - 4000	2,030	571,091.00	19.96
12	TERENGGANU	1	2500 and below	8,007	2,287,595.00	80.14
		2	5501 - 7000	2,016	566,889.50	19.86
13	NEGERI SEMBILAN	1	4001 - 5500	4,038	1,147,446.50	40.23
		2	2501 - 4000	1,959	573,469.00	20.11
		3	2500 and below	2,031	565,993.50	19.84
		4	7001 - 9000	1,983	565,429.50	19.82
14	JOHOR	1	2501 - 4000	7,971	2,274,890.50	80.30
		2	2500 and below	2,031	558,201.00	19.70
15	PERAK	1	2501 - 4000	2,061	579,069.50	20.45
		2	4001 - 5500	1,988	569,305.50	20.10
		3	5501 - 7000	1,990	563,547.00	19.90
		4	7001 - 9000	2,012	563,190.00	19.89
		5	2500 and below	1,963	557,012.50	19.67
16	KEDAH	1	2501 - 4000	5,988	1,670,176.50	74.09
		2	5501 - 7000	2,042	583,981.00	25.91

63 rows selected.


Total Orders for each income group
34358
42274
36025
44349
44093
2500 and below
2501 - 4000
4001 - 5500
5501 - 7000
7001 - 9000



The pie chart illustrates the total number of orders across six income groups, showing a fairly balanced distribution. The income group 4001 - 5500 leads with the highest number of orders at 44,093, followed by 2501 - 4000 with 42,274. Other notable groups include 5501 - 7000 with 36,025 and 2500 and below at 34,358, while the 7001 - 9000 and 2500 and below groups have the lowest order totals, both around 34,358. This suggests that middle-income brackets drive the highest number of orders. The bar chart shows the ranking of Malaysian states based on the total number of orders. Perlis ranks first, followed by Kuala Lumpur and Pahang in second and third places, respectively. Putrajaya ranks fifth, while Selangor and Kelantan are positioned lower at 10th and 9th. Johor,

Malacca, and Penang are mid-ranked, while Terengganu and Kedah have lower ranks, at 12th and 16th. The distribution reflects diverse order volumes across states, with notable performance from smaller regions like Perlis and Kuala Lumpur. Based on the two analyses I would highly recommend targeting high-performing states with tailored promotions such as promotions, bundle deals, and loyalty rewards to maintain and increase customer retention. For states ranked lower in total sales, consider offering special discounts or lower delivery fees. This can attract more customers in these areas.

3.2 Joash Alwinn Voon Dirui

3.2.1 Single Quarter Trend Analysis of Dine-In Meal Times from 2022 to 2024

SQL:

```
CREATE OR REPLACE VIEW Query1 AS
WITH cte_Meal_Time_Data AS (
    SELECT B.cal_year Years,
           B.cal_quarter Quarter,
           SUBSTR(A.orderTime, 1, 2) OrderTime,
           COUNT(A.orderID) NumberOfOrders,
           SUM(A.orderAmount) Total_Sales,
           CASE
               WHEN (SUBSTR(A.orderTime, 1, 2) IN ('09', '10', '11')) THEN 'Breakfast'
               WHEN (SUBSTR(A.orderTime, 1, 2) IN ('12', '13', '14')) THEN 'Lunch'
               WHEN (SUBSTR(A.orderTime, 1, 2) IN ('15', '16', '17')) THEN 'Tea Time'
               ELSE 'Dinner'
           END Meal_Time
    FROM orders_fact    A
    JOIN date_dim       B ON A.date_key = B.date_key
    JOIN customer_dim   C ON A.customer_key = C.customer_key
    WHERE A.status = 'DINE-IN'
    AND B.cal_year BETWEEN 2022 AND 2024
    GROUP BY B.cal_year, B.cal_quarter, SUBSTR(A.orderTime, 1, 2)
    ORDER BY Meal_Time
),
cte_Trend_Analysis AS (
    SELECT Years,
           Quarter,
           Meal_Time,
           SUM(NumberOfOrders) NumberOfOrders,
           SUM(Total_Sales) AS Total_Sales,
```

```
            LAG(SUM(Total_Sales), 1) OVER (PARTITION BY Meal_Time ORDER BY Years, Quarter) AS Prev_Quarter_Sales,
            LAG(SUM(NumberOfOrders), 1) OVER (PARTITION BY Meal_Time ORDER BY Years, Quarter) AS
Prev_No_Of_Orders
    FROM cte_Meal_Time_Data
    GROUP BY Years, Quarter, Meal_Time
),
cte_Trend_Classification AS (
    SELECT Years,
           Quarter,
           Meal_Time,
           NumberOfOrders,
           Prev_No_Of_Orders,
           Total_Sales,
           Prev_Quarter_Sales,
           COALESCE(Total_Sales - Prev_Quarter_Sales, 0) AS Sales_Change,
           CASE
               WHEN Prev_Quarter_Sales IS NULL THEN 'N/A' -- First quarter data
               WHEN Total_Sales > Prev_Quarter_Sales THEN 'Increasing'
               WHEN Total_Sales < Prev_Quarter_Sales THEN 'Decreasing'
               ELSE 'Stable'
           END AS Sales_Trend,
           CASE
               WHEN Prev_No_Of_Orders IS NULL OR Prev_No_Of_Orders = 0 THEN 0
               ELSE ROUND(((NumberOfOrders - Prev_No_Of_Orders) / Prev_No_Of_Orders) * 100, 2)
           END AS Orders_Change,
           CASE
               WHEN Prev_No_Of_Orders IS NULL THEN 'N/A' -- First quarter data
               WHEN NumberOfOrders > Prev_No_Of_Orders THEN 'Increasing'
               WHEN NumberOfOrders < Prev_No_Of_Orders THEN 'Decreasing'
               ELSE 'Stable'
           END AS Orders_Trend
    FROM cte_Trend_Analysis
),
cte_Avg_Percentage_Change AS (
```

```
    SELECT Years,
           Quarter,
           Meal_Time,
           NumberOfOrders,
           Orders_Change,
           Orders_Trend,
           Total_Sales,
           Sales_Change,
           Sales_Trend
    FROM cte_Trend_Classification
)
SELECT Years,
       Quarter,
       Meal_Time,
       NumberOfOrders,
       Orders_Change,
       Orders_Trend,
       Total_Sales,
       Sales_Change,
       Sales_Trend
FROM cte_Avg_Percentage_Change
ORDER BY Meal_Time, Years, Quarter;

CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF

SET LINESIZE 84
SET PAGESIZE 40
ALTER SESSION SET NLS_DATE_FORMAT = 'dd/mm/yyyy';


ACCEPT v_quarter VARCHAR(2) PROMPT 'Enter Quarter for the analysis: '
```

```
TTITLE CENTER 'Single Quarter Trend Analysis of Dine-In Meal Times from 2022 to 2024' SKIP 2 -
LEFT 'Date Generated: ' _DATE -
RIGHT 'Page: ' FORMAT 999 SQL.PNO SKIP 2
BREAK ON Meal_Time SKIP 1 ON Quarter
COLUMN Years FORMAT 9999 HEADING "Years";
COLUMN Quarter FORMAT A7 HEADING "Quarter";
COLUMN Meal_Time FORMAT A9 HEADING "Meal|Time";
COLUMN NumberOfOrders FORMAT 9,999 HEADING "Number|Of|Orders";
COLUMN Orders_Change FORMAT 99.99 HEADING "Orders|Change|(%)";
COLUMN Orders_Trend FORMAT A10 HEADING "Orders|Trend";
COLUMN Total_Sales FORMAT $999,999.99 HEADING "Total|Sales";
COLUMN Sales_Change FORMAT $99,999.99 HEADING "Sales|Change";
COLUMN Sales_Trend FORMAT A10 HEADING "Sales|Trend";

COMPUTE AVG LABEL ' Average: ' OF NumberOfOrders ON Meal_Time
COMPUTE AVG LABEL ' Average: ' OF Orders_Change ON Meal_Time
COMPUTE AVG LABEL ' Average: ' OF Total_Sales ON Meal_Time
COMPUTE AVG LABEL ' Average: ' OF Sales_Change ON Meal_Time

select *
from Query1
where Quarter = '&v_quarter';
```

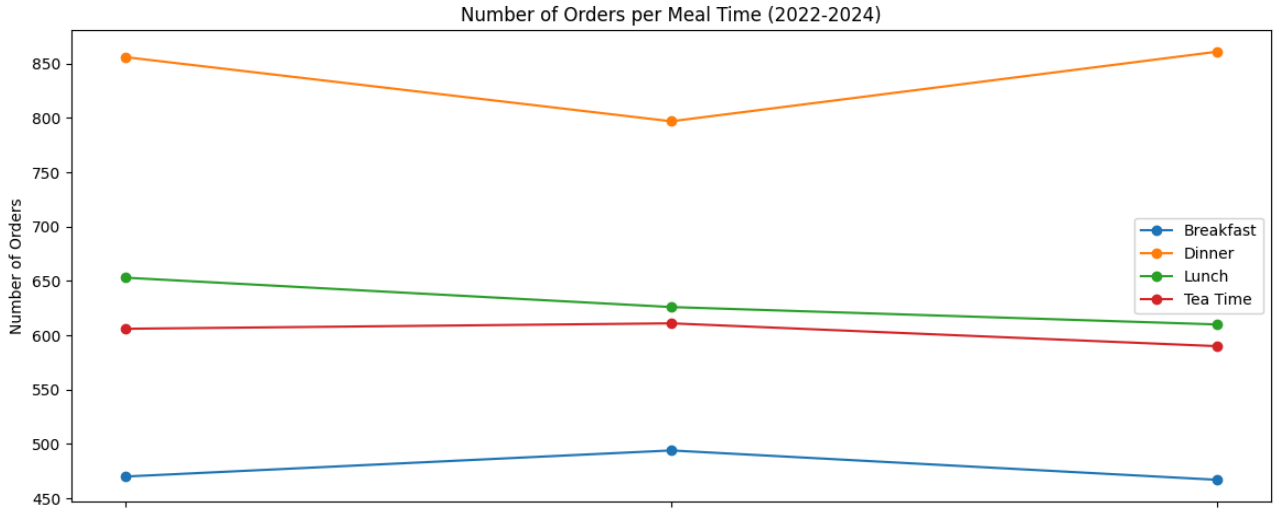
OUTPUT:

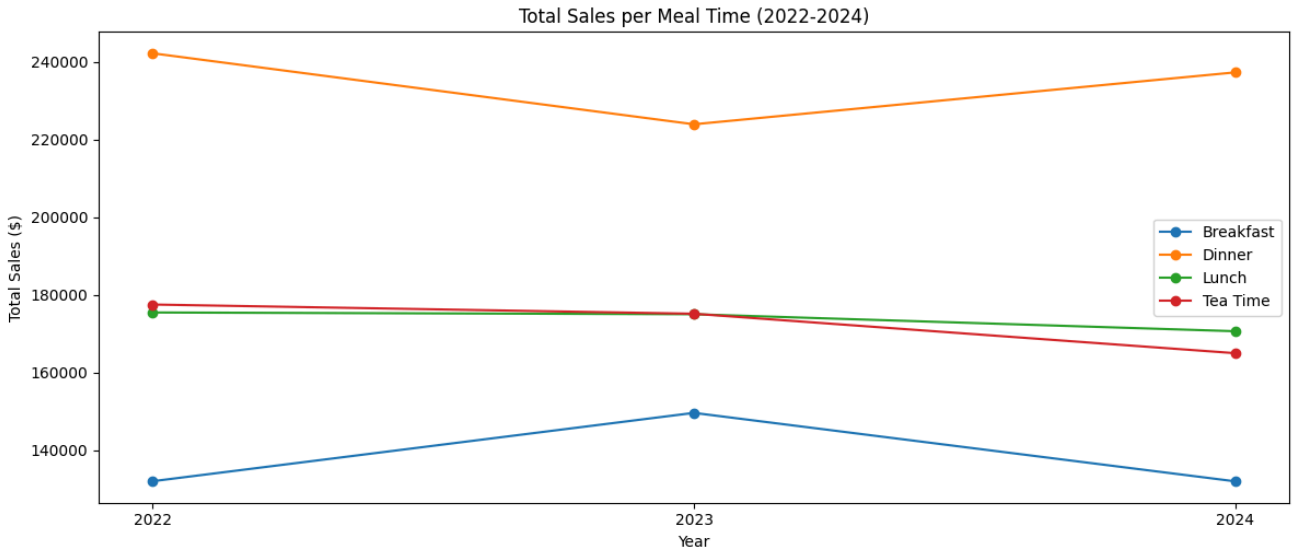
Single Quarter Trend Analysis of Dine-In Meal Times from 2022 to 2024

Date Generated: 21/09/2024 Page: 1

Years	Quarter	Meal Time	Number Of Orders	Orders Change (%)	Orders Trend	Total Sales	Sales Change	Sales Trend
2022	Q4	Breakfast	470	-9.44	Decreasing	$132,042.50	-$14,222.50	Decreasing
2023			494	-.80	Decreasing	$149,638.00	$16,137.00	Increasing
2024			467	-1.27	Decreasing	$132,011.50	-$2,563.50	Decreasing
	*******	*********	------	------		------------	-----------	
		Average:	477	-3.84		$137,897.33	-$216.33	
2022	Q4	Dinner	856	7.81	Increasing	$242,243.50	$19,606.50	Increasing
2023			797	-4.21	Decreasing	$223,967.50	$3,732.50	Increasing
2024			861	2.50	Increasing	$237,333.50	-$1,047.50	Decreasing
	*******	*********	------	------		------------	-----------	
		Average:	838	2.03		$234,514.83	$7,430.50	
2022	Q4	Lunch	653	15.99	Increasing	$175,496.50	$25,818.00	Increasing
2023			626	4.51	Increasing	$175,047.00	$2,894.50	Increasing
2024			610	-.81	Decreasing	$170,669.00	$91.50	Increasing
	*******	*********	------	------		------------	-----------	
		Average:	630	6.56		$173,737.50	$9,601.33	

2022 Q4	Tea Time	606	4.12	Increasing	$177,543.50	$6,983.50	Increasing
2023		611	.49	Increasing	$175,172.00	-$2,221.50	Decreasing
2024		590	-8.81	Decreasing	$165,016.00	-$19,935.00	Decreasing
*******	*********	------	------		----------	----------	
	Average:	602	-1.40		$172,577.17	-$5,057.67	





This quarterly trend analysis of dine-in meal times from 2022 to 2024 highlights patterns in orders and sales, helping allocate staff and manage inventory more efficiently. For instance, decreasing trends in breakfast orders suggest fewer resources might be needed, while increasing dinner orders point to the need for more staff. To forecast for 2025, you can use the average order and sales changes. For example, applying the average order change of -3.84% to breakfast gives an estimated 449 orders, and adjusting sales by the average change results in projected sales of $137,681. Therefore, based on the forecasting for Breakfast in Q4 in 2025, since the decrease in the number of orders is not significant, the amount of staff planning and inventory management can be maintained. These projections help plan staffing and inventory for future demand.

3.2.2 Stall and Dish Revenue Contribution Analysis for Underperforming Stalls by Year

SQL:

```
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF

SET LINESIZE 142
SET PAGESIZE 40
ALTER SESSION SET NLS_DATE_FORMAT = 'dd/mm/yyyy';

ACCEPT v_year NUMBER(4) PROMPT 'Enter Year for the analysis: '

TTITLE CENTER 'Top 50% Items Ordered for Underperforming Stalls Based on Stall Contribution by Year'
SKIP 2 -
LEFT 'Date Generated: ' _DATE -
RIGHT 'Page: ' FORMAT 999 SQL.PNO SKIP 2
BREAK ON stallName SKIP 1 ON YEAR_TOTAL_SALES ON STALL_CONTRIBUTION_PERCENT

COLUMN stallName FORMAT A25 HEADING "Stall Name"
COLUMN itemName FORMAT A25 HEADING "Item Name"
COLUMN Q1_SALES FORMAT $99,999 HEADING "Q1|Sales"
COLUMN Q2_SALES FORMAT $99,999 HEADING "Q2|Sales"
COLUMN Q3_SALES FORMAT $99,999 HEADING "Q3|Sales"
COLUMN Q4_SALES FORMAT $99,999 HEADING "Q4|Sales"
COLUMN Q1_QTY FORMAT 9,999 HEADING "Q1|Qty"
COLUMN Q2_QTY FORMAT 9,999 HEADING "Q2|Qty"
COLUMN Q3_QTY FORMAT 9,999 HEADING "Q3|Qty"
COLUMN Q4_QTY FORMAT 9,999 HEADING "Q4|Qty"
COLUMN Dish_TOTAL_SALES FORMAT $999,999 HEADING "Dish|Total|Sales"
COLUMN YEAR_TOTAL_SALES FORMAT $999,999 HEADING "Stall|Total|Sales"
COLUMN STALL_CONTRIBUTION_PERCENT FORMAT 99.99 HEADING "Stall|Cont|%"
```

```
COMPUTE AVG LABEL ' Average: ' OF Q1_SALES ON stallName
COMPUTE AVG LABEL ' Average: ' OF Q2_SALES ON stallName
COMPUTE AVG LABEL ' Average: ' OF Q3_SALES ON stallName
COMPUTE AVG LABEL ' Average: ' OF Q4_SALES ON stallName
COMPUTE AVG LABEL ' Average: ' OF Q1_QTY ON stallName
COMPUTE AVG LABEL ' Average: ' OF Q2_QTY ON stallName
COMPUTE AVG LABEL ' Average: ' OF Q3_QTY ON stallName
COMPUTE AVG LABEL ' Average: ' OF Q4_QTY ON stallName
COMPUTE AVG LABEL ' Average: ' OF Dish_TOTAL_SALES ON stallName
COMPUTE AVG LABEL ' Average: ' OF DISH_CONTRIBUTION_PERCENT ON stallName

WITH cte_Pivot_Sales AS (
    SELECT
        B.cal_quarter,
        D.stallName,
        D.itemName,
        SUM(A.lineTotal + CASE WHEN E.deliveryFee > 0 THEN E.deliveryFee ELSE 0 END) AS Total_Sales,
        SUM(A.quantity) AS Total_Qty
    FROM items_Fact A
    JOIN date_dim B ON A.date_key = B.date_key
    JOIN customer_dim C ON A.customer_key = C.customer_key
    JOIN menu_dim D ON A.menu_key = D.menu_key
    JOIN Orders_Fact E ON A.orderID = E.orderID
    WHERE B.cal_year = '&v_year'
    GROUP BY B.cal_quarter, D.stallName, D.itemName
),
cte_Pivoted_Sales AS (
    SELECT * FROM (
        SELECT cal_quarter, stallName, itemName, Total_Sales, Total_Qty
        FROM cte_Pivot_Sales
    )
    PIVOT (
        SUM(Total_Sales) AS Sales, SUM(Total_Qty) AS Qty
```

```
        FOR cal_quarter IN ('Q1' AS Q1, 'Q2' AS Q2, 'Q3' AS Q3, 'Q4' AS Q4)
    )
),
cte_Stall_Total_Sales AS (
    SELECT stallName,
           SUM(Q1_Sales + Q2_Sales + Q3_Sales + Q4_Sales) AS YEAR_TOTAL_SALES
    FROM cte_Pivoted_Sales
    GROUP BY stallName
),
cte_Total_Revenue AS (
    SELECT SUM(YEAR_TOTAL_SALES) AS TOTAL_REVENUE
    FROM cte_Stall_Total_Sales
),
Ranked_Sales AS (
    SELECT
        A.stallName,
        A.itemName,
        A.Q1_Sales, A.Q2_Sales, A.Q3_Sales, A.Q4_Sales,
        A.Q1_Qty, A.Q2_Qty, A.Q3_Qty, A.Q4_Qty,
        (A.Q1_Sales + A.Q2_Sales + A.Q3_Sales + A.Q4_Sales) AS DISH_TOTAL_SALES,
        B.YEAR_TOTAL_SALES,
        (B.YEAR_TOTAL_SALES / C.TOTAL_REVENUE) * 100 AS STALL_CONTRIBUTION_PERCENT,
        PERCENT_RANK() OVER (PARTITION BY A.stallName ORDER BY A.Q1_Sales + A.Q2_Sales + A.Q3_Sales +
A.Q4_Sales DESC) AS PercentRank
    FROM cte_Pivoted_Sales A
    JOIN cte_Stall_Total_Sales B ON A.stallName = B.stallName
    CROSS JOIN cte_Total_Revenue C
)
SELECT stallName,
       itemName,
       Q1_Sales, Q2_Sales, Q3_Sales, Q4_Sales,
       Q1_Qty, Q2_Qty, Q3_Qty, Q4_Qty,
       DISH_TOTAL_SALES,
       YEAR_TOTAL_SALES,
```

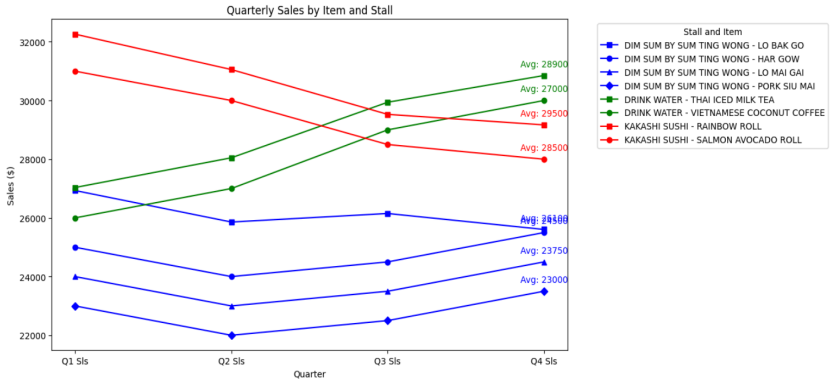
```
       STALL_CONTRIBUTION_PERCENT
FROM Ranked_Sales
WHERE STALL_CONTRIBUTION_PERCENT <= 8
AND PercentRank <= 0.5
ORDER BY stallName, STALL_CONTRIBUTION_PERCENT DESC;
```

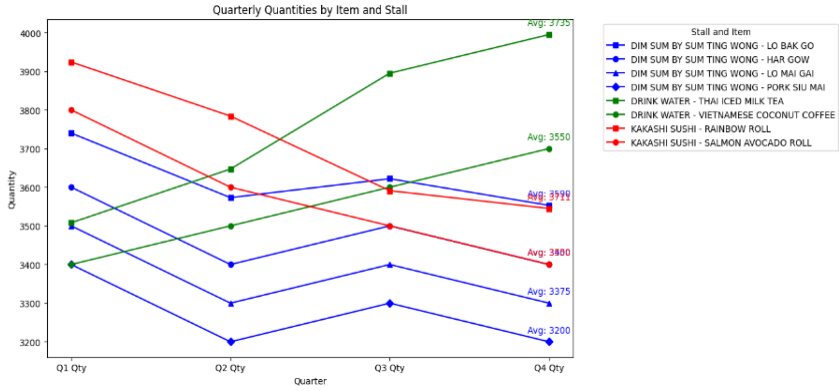
OUTPUT:

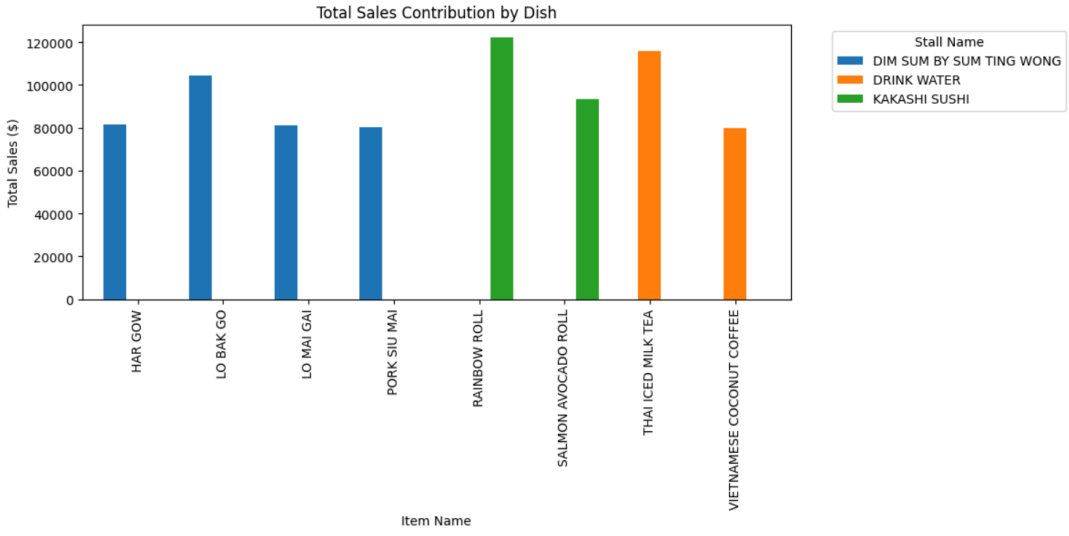
Top 50% Items Ordered for Underperforming Stalls Based on Stall Contribution by Year

Date Generated: 21/09/2024 Page: 1

Stall Name	Item Name	Q1 Sales	Q2 Sales	Q3 Sales	Q4 Sales	Q1 Qty	Q2 Qty	Q3 Qty	Q4 Qty	Dish Total Sales	Stall Total Sales	Stall Cont %
DIM SUM BY SUM TING WONG	LO BAK GO	$26,933	$25,859	$26,150	$25,606	3,740	3,573	3,622	3,553	$104,547	$398,727	6.86
	HAR GOW	$19,111	$18,657	$20,810	$23,119	3,641	3,588	3,993	4,444	$81,697		
	LO MAI GAI	$20,536	$17,440	$21,158	$21,883	3,946	3,321	4,068	4,191	$81,017		
**********		--------	--------	--------	--------	------	------	------	------	---------	*********	******
Average:		$22,193	$20,652	$22,706	$23,536	3,776	3,494	3,894	4,063	$89,087		
DRINK WATER	THAI ICED MILK TEA	$27,035	$28,048	$29,941	$30,852	3,508	3,647	3,895	3,995	$115,875	$393,646	6.77
	VIETNAMESE COCONUT COFFEE	$19,788	$20,468	$21,338	$18,473	3,794	3,927	4,103	3,550	$80,068		
	CALAMANSI LIMEADE	$18,306	$21,172	$20,306	$18,233	3,513	4,057	3,892	3,505	$78,018		
**********		--------	--------	--------	--------	------	------	------	------	---------	*********	******
Average:		$21,710	$23,229	$23,862	$22,519	3,605	3,877	3,963	3,683	$91,320		
KAKASHI SUSHI	RAINBOW ROLL	$32,262	$31,055	$29,528	$29,169	3,924	3,784	3,591	3,545	$122,014	$447,640	7.70
	SALMON AVOCADO ROLL	$23,750	$21,569	$23,629	$24,401	3,836	3,471	3,810	3,931	$93,350		
	CALIFORNIA ROLL	$20,656	$22,032	$22,386	$19,997	3,966	4,242	4,300	3,859	$85,070		
**********		--------	--------	--------	--------	------	------	------	------	---------	*********	******
Average:		$25,556	$24,885	$25,181	$24,522	3,909	3,832	3,900	3,778	$100,145		


Quarterly Sales by Item and Stall
Sales ($)
Quarter
Q1 Sls
Q2 Sls
Q3 Sls
Q4 Sls
Avg: 28900
Avg: 27000
Avg: 29500
Avg: 28500
Avg: 23750
Avg: 23000
Stall and Item
DIM SUM BY SUM TING WONG - LO BAK GO
DIM SUM BY SUM TING WONG - HAR GOW
DIM SUM BY SUM TING WONG - LO MAI GAI
DIM SUM BY SUM TING WONG - PORK SIU MAI
DRINK WATER - THAI ICED MILK TEA
DRINK WATER - VIETNAMESE COCONUT COFFEE
KAKASHI SUSHI - RAINBOW ROLL
KAKASHI SUSHI - SALMON AVOCADO ROLL


Quarterly Quantities by Item and Stall
Quantity
Quarter
Q1 Qty
Q2 Qty
Q3 Qty
Q4 Qty
Avg: 3735
Avg: 3550
Avg: 3375
Avg: 3200
Stall and Item
DIM SUM BY SUM TING WONG - LO BAK GO
DIM SUM BY SUM TING WONG - HAR GOW
DIM SUM BY SUM TING WONG - LO MAI GAI
DIM SUM BY SUM TING WONG - PORK SIU MAI
DRINK WATER - THAI ICED MILK TEA
DRINK WATER - VIETNAMESE COCONUT COFFEE
KAKASHI SUSHI - RAINBOW ROLL
KAKASHI SUSHI - SALMON AVOCADO ROLL


Total Sales Contribution by Dish
Total Sales ($)
Item Name
HAR GOW
LO BAK GO
LO MAI GAI
PORK SIU MAI
RAINBOW ROLL
SALMON AVOCADO ROLL
THAI ICED MILK TEA
VIETNAMESE COCONUT COFFEE
Stall Name
DIM SUM BY SUM TING WONG
DRINK WATER
KAKASHI SUSHI

This query identifies underperforming stalls that contribute less than 8% of the total revenue at the Hawker Centre, such as “DIM SUM BY SUM TING WONG” (6.86%) and “KAKASHI SUSHI” (7.70%). Within these stalls, it highlights specific menu items that contribute top 50% items ordered for each stall, such as “LO BAK GO”, “HAR GAO”, “LO MAI GAI” for “DIM SUM BY SUM TING WONG”. By analysing these items, you can focus promotional efforts on them to boost overall sales. For example, the hawker centre can offer a “Buy 1 LO BAK GO, Get 50% Off HAR GOW” promotion for two weeks. This combo deal could encourage customers to try more of the high-performing dishes. The promotion should be available during peak hours (e.g., lunch and dinner), where foot traffic is higher, and should be capped at 100 redemptions per day to control cost. Not only that, the quarterly trend analysis provides insight into how these items perform over the year, guiding decisions about when to apply promotions, optimise inventory, and adjust staff allocation based on peak and low-demand periods.

3.2.3 Popular Food Items Based On Customers City of Original State

SQL:

```
CREATE OR REPLACE VIEW Query3 AS
WITH Ranked_Items AS (
    SELECT C.state AS State,
           C.city AS City,
           D.itemName AS Item_Name,
           SUM(A.quantity) AS Total_Quantity,
           COUNT(DISTINCT B.orderID) AS Total_Orders,
           SUM(A.lineTotal) AS Total_Spending,
           PERCENT_RANK() OVER (PARTITION BY C.city ORDER BY SUM(A.LINETOTAL) DESC) AS
PercentRank
    FROM Items_Fact    A
    JOIN Orders_Fact   B ON A.orderID = B.orderID
    JOIN customer_dim  C ON A.customer_key = C.customer_key
    JOIN menu_dim      D ON A.menu_key = D.menu_key
    GROUP BY C.state, C.city, D.itemName
)
SELECT State,
       City,
       Item_Name,
       Total_Quantity,
       Total_Orders,
       Total_Spending
FROM Ranked_Items
WHERE PercentRank <= 0.20
ORDER BY City, Total_Quantity DESC;

CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
```

```
TTITLE OFF

SET LINESIZE 98
SET PAGESIZE 40
ALTER SESSION SET NLS_DATE_FORMAT = 'dd/mm/yyyy';

ACCEPT v_state CHAR(20) PROMPT 'Enter the State for analysis: '

TTITLE CENTER 'Popular Food Items Based On Customers City of Original State' SKIP 2 -
LEFT 'Date Generated: ' _DATE -
RIGHT 'Page: ' FORMAT 999 SQL.PNO SKIP 2
BREAK ON State ON City SKIP 1
COLUMN State FORMAT A16 HEADING "State";
COLUMN City FORMAT A16 HEADING "City";
COLUMN Item_Name FORMAT A35 HEADING "Item Name";
COLUMN Total_Quantity FORMAT 99,999 HEADING "Total|Quantity";
COLUMN Total_Orders FORMAT 9,999 HEADING "Total|Orders";
COLUMN Total_Spending FORMAT $999,999.99 HEADING "Total|Spending";

COMPUTE SUM LABEL ' Total: ' OF Total_Quantity ON city
COMPUTE SUM LABEL ' Total: ' OF Total_Orders ON city
COMPUTE SUM LABEL ' Total: ' OF Total_Spending ON city


select *
from Query3
where State = '&v_state';
```

OUTPUT:

Popular Food Items Based On Customers City of Original State

Date Generated: 21/09/2024 Page: 1

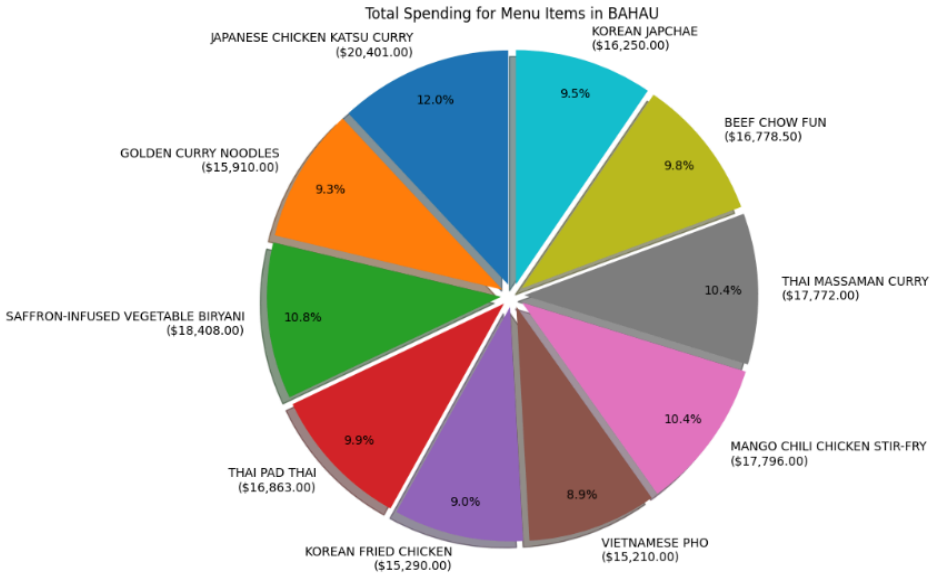
State	City	Item Name	Total Quantity	Total Orders	Total Spending
NEGERI SEMBILAN	BAHAU	JAPANESE CHICKEN KATSU CURRY	1,774	126	$20,401.00
		GOLDEN CURRY NOODLES	1,591	120	$15,910.00
		SAFFRON-INFUSED VEGETABLE BIRYANI	1,534	125	$18,408.00
		THAI PAD THAI	1,533	111	$16,863.00
		KOREAN FRIED CHICKEN	1,529	106	$15,290.00
		VIETNAMESE PHO	1,521	111	$15,210.00
		MANGO CHILI CHICKEN STIR-FRY	1,483	109	$17,796.00
		THAI MASSAMAN CURRY	1,481	105	$17,772.00
		BEEF CHOW FUN	1,459	106	$16,778.50
		KOREAN JAPCHAE	1,250	94	$16,250.00
	**********				
	Total:		15,155	1,113	$170,678.50
	NILAI	MANGO CHILI CHICKEN STIR-FRY	3,260	248	$39,120.00
		BEEF CHOW FUN	3,239	242	$37,248.50
		GOLDEN CURRY NOODLES	3,230	251	$32,300.00
		SARAWAK LAKSA	3,228	248	$32,280.00
		JAPANESE RAMEN	3,215	259	$35,365.00
		INDIAN CHICKEN TIKKA MASALA	3,065	242	$32,182.50
		KOREAN JAPCHAE	2,984	231	$38,792.00
		JAPANESE CHICKEN KATSU CURRY	2,927	239	$33,660.50
		SAFFRON-INFUSED VEGETABLE BIRYANI	2,883	230	$34,596.00

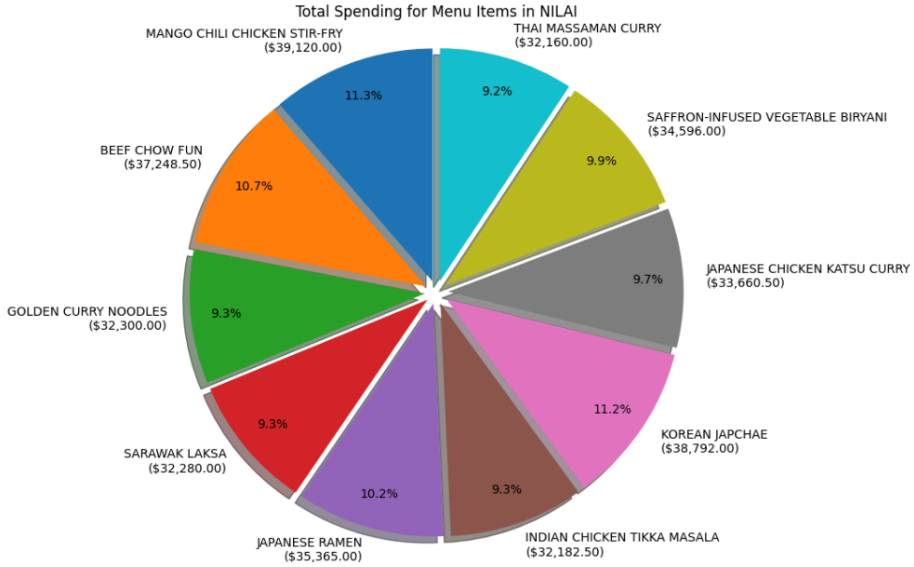
		THAI MASSAMAN CURRY	2,680	213	$32,160.00
	**********		--------	------	----------
	Total:		30,711	2,403	$347,704.50
	SI RUSA	SAFFRON-INFUSED VEGETABLE BIRYANI	3,804	275	$45,648.00
		KOREAN FRIED CHICKEN	3,446	248	$34,460.00
		VIETNAMESE PHO	3,394	258	$33,940.00
		INDIAN CHICKEN TIKKA MASALA	3,362	251	$35,301.00
		KOREAN JAPCHAE	3,316	253	$43,108.00
		SARAWAK LAKSA	3,234	249	$32,340.00

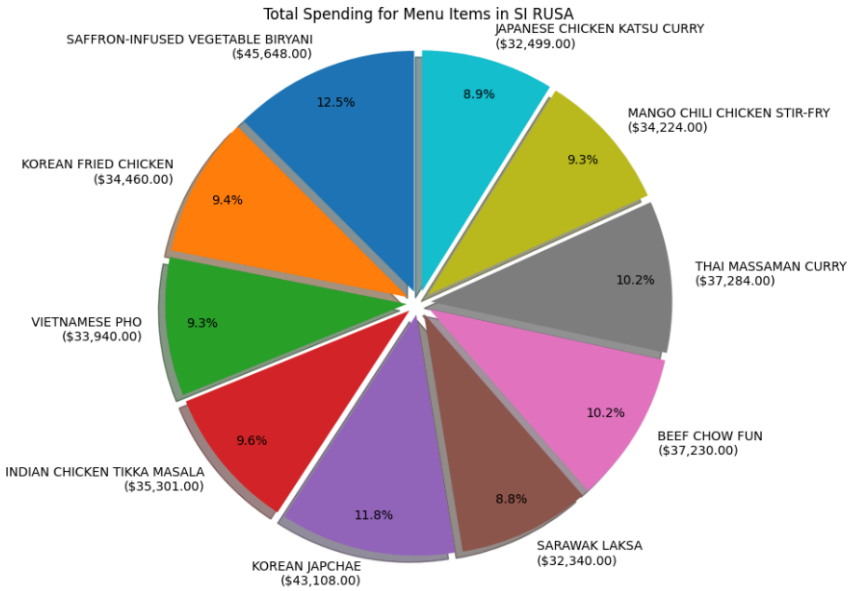
Popular Food Items Based On Customers City of Original State

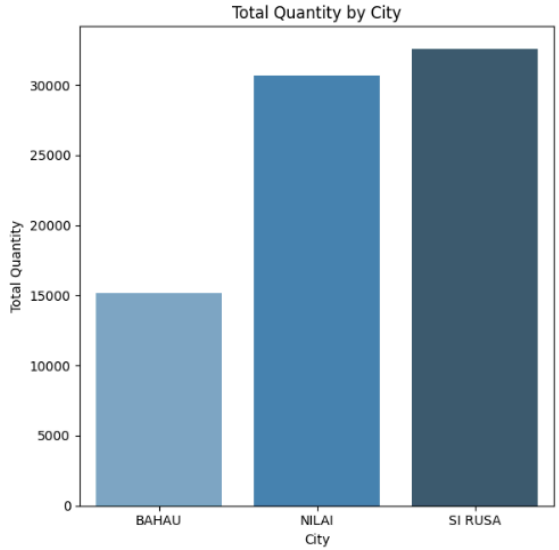
Date Generated: 21/09/2024 Page: 2

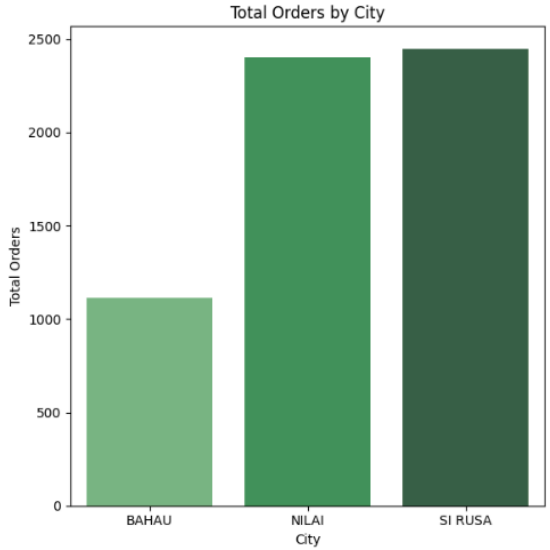
State	City	Item Name	Total Quantity	Total Orders	Total Spending
NEGERI SEMBILAN	SI RUSA	BEEF CHOW FUN	3,220	235	$37,030.00
		THAI MASSAMAN CURRY	3,107	225	$37,284.00
		MANGO CHILI CHICKEN STIR-FRY	2,852	224	$34,224.00
		JAPANESE CHICKEN KATSU CURRY	2,826	228	$32,499.00
	**********		--------	------	----------
	Total:		32,561	2,446	$365,834.00

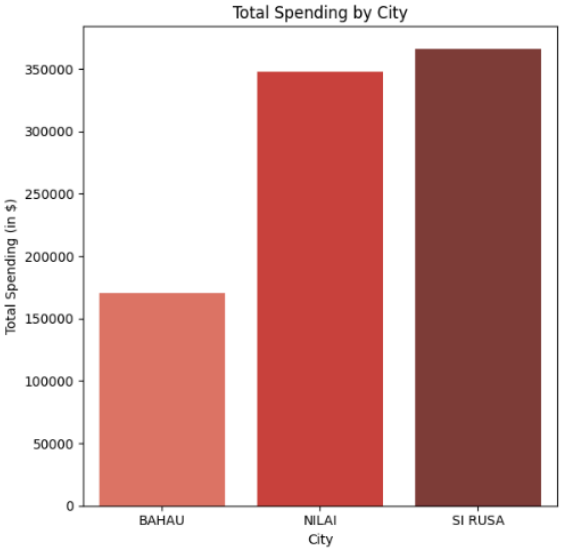

Total Spending for Menu Items in BAHAU
JAPANESE CHICKEN KATSU CURRY ($20,401.00)
12.0%
KOREAN JAPCHAE ($16,250.00)
9.5%
BEEF CHOW FUN ($16,778.50)
9.8%
GOLDEN CURRY NOODLES ($15,910.00)
9.3%
THAI MASSAMAN CURRY ($17,772.00)
10.4%
SAFFRON-INFUSED VEGETABLE BIRYANI ($18,408.00)
10.8%
MANGO CHILI CHICKEN STIR-FRY ($17,796.00)
10.4%
THAI PAD THAI ($16,863.00)
9.9%
VIETNAMESE PHO ($15,210.00)
8.9%
KOREAN FRIED CHICKEN ($15,290.00)
9.0%


Total Spending for Menu Items in NILAI
MANGO CHILI CHICKEN STIR-FRY ($39,120.00)
11.3%
THAI MASSAMAN CURRY ($32,160.00)
9.2%
SAFFRON-INFUSED VEGETABLE BIRYANI ($34,596.00)
9.9%
BEEF CHOW FUN ($37,248.50)
10.7%
JAPANESE CHICKEN KATSU CURRY ($33,660.50)
9.7%
GOLDEN CURRY NOODLES ($32,300.00)
9.3%
KOREAN JAPCHAE ($38,792.00)
11.2%
SARAWAK LAKSA ($32,280.00)
9.3%
INDIAN CHICKEN TIKKA MASALA ($32,182.50)
9.3%
JAPANESE RAMEN ($35,365.00)
10.2%


Total Spending for Menu Items in SI RUSA
SAFFRON-INFUSED VEGETABLE BIRYANI ($45,648.00)
12.5%
JAPANESE CHICKEN KATSU CURRY ($32,499.00)
8.9%
MANGO CHILI CHICKEN STIR-FRY ($34,224.00)
9.3%
KOREAN FRIED CHICKEN ($34,460.00)
9.4%
THAI MASSAMAN CURRY ($37,284.00)
10.2%
VIETNAMESE PHO ($33,940.00)
9.3%
BEEF CHOW FUN ($37,230.00)
10.2%
INDIAN CHICKEN TIKKA MASALA ($35,301.00)
9.6%
SARAWAK LAKSA ($32,340.00)
8.8%
KOREAN JAPCHAE ($43,108.00)
11.8%


Total Quantity by City
Total Quantity
0
5000
10000
15000
20000
25000
30000
BAHAU
NILAI
SI RUSA
City


Total Orders by City
Total Orders
0
500
1000
1500
2000
2500
BAHAU
NILAI
SI RUSA
City


Total Spending by City
Total Spending (in $)
0
50000
100000
150000
200000
250000
300000
350000
BAHAU
NILAI
SI RUSA
City

This analysis identifies the “Popular Food Items Based On Customers City of Original State”, such as "Mango Chili Chicken Stir-Fry" in Nilai and "Saffron-Infused Vegetable Biryani" in Si Rusa. By examining total orders, item quantity, and spending per city, businesses can make informed decisions about expanding to high-demand areas. For example, with Customers from Nilai showing high sales volumes for multiple items, it could be a strong candidate for a new branch. Additionally, these insights allow us to forecast potential earnings based on the sum of the total spending of the top 50% menu items in the city. Not only that, by knowing the sum of the total orders and the quantity, this allows staff planning and inventory management for the new branch that is to come.

3.3 Loh Jia Shou

3.3.1 Daily Orders by Meal Type Between The Year 2022 To 2024

SQL:

```
CL SCR

-- Set pagesize and linesize
SET LINESIZE 120
SET PAGESIZE 26

-- Format
COLUMN Day              FORMAT A13          HEADING "DAY"
COLUMN Meal_Type        FORMAT A15          HEADING "MEAL TYPE"
COLUMN day_of_week      FORMAT '9'          HEADING " "
COLUMN TotalOrders2022  FORMAT '99,999'     HEADING "2022    | TOTAL ORDERS"
COLUMN TotalOrders2023  FORMAT '99,999'     HEADING "2023    | TOTAL ORDERS"
COLUMN TotalOrders2024  FORMAT '99,999'     HEADING "2024    | TOTAL ORDERS"
COLUMN AvgSalesPerc     FORMAT '99'         HEADING "AVERAGE    | SALES GROWTH %"
COLUMN AvgOrdersPerc    FORMAT '99'         HEADING "AVERAGE    | ORDERS GROWTH %"
COLUMN PredictOrders    FORMAT '99,999'     HEADING "2025     | Predicted Orders"
COLUMN OrderDiff        FORMAT '9,999'      HEADING "ORDERS  | DIFFERENCE"
BREAK ON day SKIP 1 ON day_of_week

-- Title
TTITLE      CENTER 'Daily Orders by Meal Period Between The Year 2022 To 2024' -
        SKIP 1 -
        RIGHT 'Date: ' _DATE -
        SKIP 1 -
```

```
        RIGHT 'Page No: ' FORMAT 999 SQL.PNO -
        SKIP 2 -

COMPUTE SUM LABEL 'GRAND TOTAL: '       OF TotalOrders2022   ON Day
COMPUTE SUM                             OF TotalOrders2023   ON Day
COMPUTE SUM                             OF TotalOrders2024   ON Day

CREATE OR REPLACE VIEW MealTypeView AS
SELECT
        ODF.date_key,
        ODF.orderid,
        D.day_of_week,
        D.cal_year,
        CASE
                WHEN D.day_of_week = 1 THEN 'Monday'
                WHEN D.day_of_week = 2 THEN 'Tuesday'
                WHEN D.day_of_week = 3 THEN 'Wednesday'
                WHEN D.day_of_week = 4 THEN 'Thursday'
                WHEN D.day_of_week = 5 THEN 'Friday'
                WHEN D.day_of_week = 6 THEN 'Saturday'
                WHEN D.day_of_week = 7 THEN 'Sunday'
        END AS Day,
        CASE
                WHEN ODF.ordertime BETWEEN '09:00' AND '11:00' THEN '1)Breakfast'
                WHEN ODF.ordertime BETWEEN '11:01' AND '12:00' THEN '2)Morning Tea'
                WHEN ODF.ordertime BETWEEN '12:01' AND '15:00' THEN '3)Lunch'
                WHEN ODF.ordertime BETWEEN '15:01' AND '18:00' THEN '4)Afternoon Tea'
                WHEN ODF.ordertime BETWEEN '18:01' AND '21:00' THEN '5)Dinner'
                WHEN ODF.ordertime BETWEEN '21:01' AND '22:00' THEN '6)Supper'
```

```
        END AS Meal_Type
FROM orders_fact     ODF
JOIN items_fact      ITF     ON ODF.orderid = ITF.orderid
JOIN date_dim        D       ON ODF.date_key = D.date_key
WHERE D.cal_year BETWEEN 2022 AND 2024;


CREATE OR REPLACE VIEW Orders2022 AS
SELECT
        cal_year,
        day_of_week,
        Day,
        Meal_Type,
        COUNT(DISTINCT(orderid)) AS TotalOrders2022
FROM MealTypeView
WHERE cal_year = 2022
GROUP BY cal_year, day_of_week, Day, Meal_Type;

CREATE OR REPLACE VIEW Orders2023 AS
SELECT
        cal_year,
        day_of_week,
        Day,
        Meal_Type,
        COUNT(DISTINCT(orderid)) AS TotalOrders2023
FROM MealTypeView
WHERE cal_year = 2023
GROUP BY cal_year, day_of_week, Day, Meal_Type;
```

```
CREATE OR REPLACE VIEW Orders2024 AS
SELECT
        cal_year,
        day_of_week,
        Day,
        Meal_Type,
        COUNT(DISTINCT(orderid)) AS TotalOrders2024
FROM MealTypeView
WHERE cal_year = 2024
GROUP BY cal_year, day_of_week, Day, Meal_Type;

CREATE OR REPLACE VIEW CompareView AS
SELECT
        O1.day_of_week,
        O1.Day,
        O1.Meal_Type,
        TotalOrders2022,
        TotalOrders2023,
        TotalOrders2024,
        ((TotalOrders2023 - TotalOrders2022) + (TotalOrders2024 - TotalOrders2023)) / 2 AS AvgOrdersDifference,
        (((TotalOrders2023 - TotalOrders2022) / TotalOrders2022 * 100) + ((TotalOrders2024 - TotalOrders2023) / TotalOrders2023 *
100)) / 2 AS AvgOrdersPerc
FROM        Orders2022 O1
JOIN        Orders2023 O2       ON O1.Day = O2.Day AND O1.Meal_Type = O2.Meal_Type
JOIN        Orders2024 O3       ON O1.Day = O3.Day AND O1.Meal_Type = O3.Meal_Type;



SELECT
```

```
        Day,
        Meal_Type,
        TotalOrders2022,
        TotalOrders2023,
        TotalOrders2024,
        AvgOrdersPerc,
        ((TotalOrders2024 * AvgOrdersPerc / 100) + TotalOrders2024) AS PredictOrders,
        ((TotalOrders2024 * AvgOrdersPerc / 100) + TotalOrders2024) - TotalOrders2024 AS OrderDiff
FROM CompareView
ORDER BY day_of_week, Meal_Type;

CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF
```

OUTPUT:

Daily Orders by Meal Period Between The Year 2022 To 2024

Date: 21-SEP-24
Page No: 1

DAY	MEAL TYPE	2022 TOTAL ORDERS	2023 TOTAL ORDERS	2024 TOTAL ORDERS	AVERAGE ORDERS GROWTH %	2025 Predicted Orders	ORDERS DIFFERENCE
Monday	1)Breakfast	357	422	423	9	462	39
	2)Morning Tea	253	268	285	6	302	17
	3)Lunch	762	973	849	7	912	63
	4)Afternoon Tea	834	914	848	1	858	10
	5)Dinner	781	891	944	10	1,039	95
	6)Supper	242	314	296	12	332	36
**********		----------	----------	----------			
GRAND TOTAL:		3,229	3,782	3,645			
Tuesday	1)Breakfast	352	410	455	14	517	62
	2)Morning Tea	228	304	267	11	295	28
	3)Lunch	705	896	921	15	1,059	138
	4)Afternoon Tea	718	856	869	10	959	90
	5)Dinner	703	847	872	12	974	102
	6)Supper	219	275	267	11	297	30
**********		----------	----------	----------			
GRAND TOTAL:		2,925	3,588	3,651			

Daily Orders by Meal Period Between The Year 2022 To 2024

Date: 21-SEP-24
Page No: 2

DAY	MEAL TYPE	2022 TOTAL ORDERS	2023 TOTAL ORDERS	2024 TOTAL ORDERS	AVERAGE ORDERS GROWTH %	2025 Predicted Orders	ORDERS DIFFERENCE
Wednesday	1)Breakfast	377	426	400	3	414	14
	2)Morning Tea	233	296	285	12	318	33
	3)Lunch	749	906	915	11	1,015	100
	4)Afternoon Tea	782	877	852	5	892	40
	5)Dinner	761	839	836	5	877	41
	6)Supper	241	280	294	11	325	31
**********		----------	----------	----------			
GRAND TOTAL:		3,143	3,624	3,582			
Thursday	1)Breakfast	306	431	410	18	484	74
	2)Morning Tea	243	267	293	10	322	29
	3)Lunch	701	899	848	11	944	96
	4)Afternoon Tea	710	886	881	12	988	107
	5)Dinner	688	887	830	11	923	93
	6)Supper	211	293	297	20	357	60
**********		----------	----------	----------			
GRAND TOTAL:		2,859	3,663	3,559			

Daily Orders by Meal Period Between The Year 2022 To 2024

Date: 21-SEP-24
Page No: 3

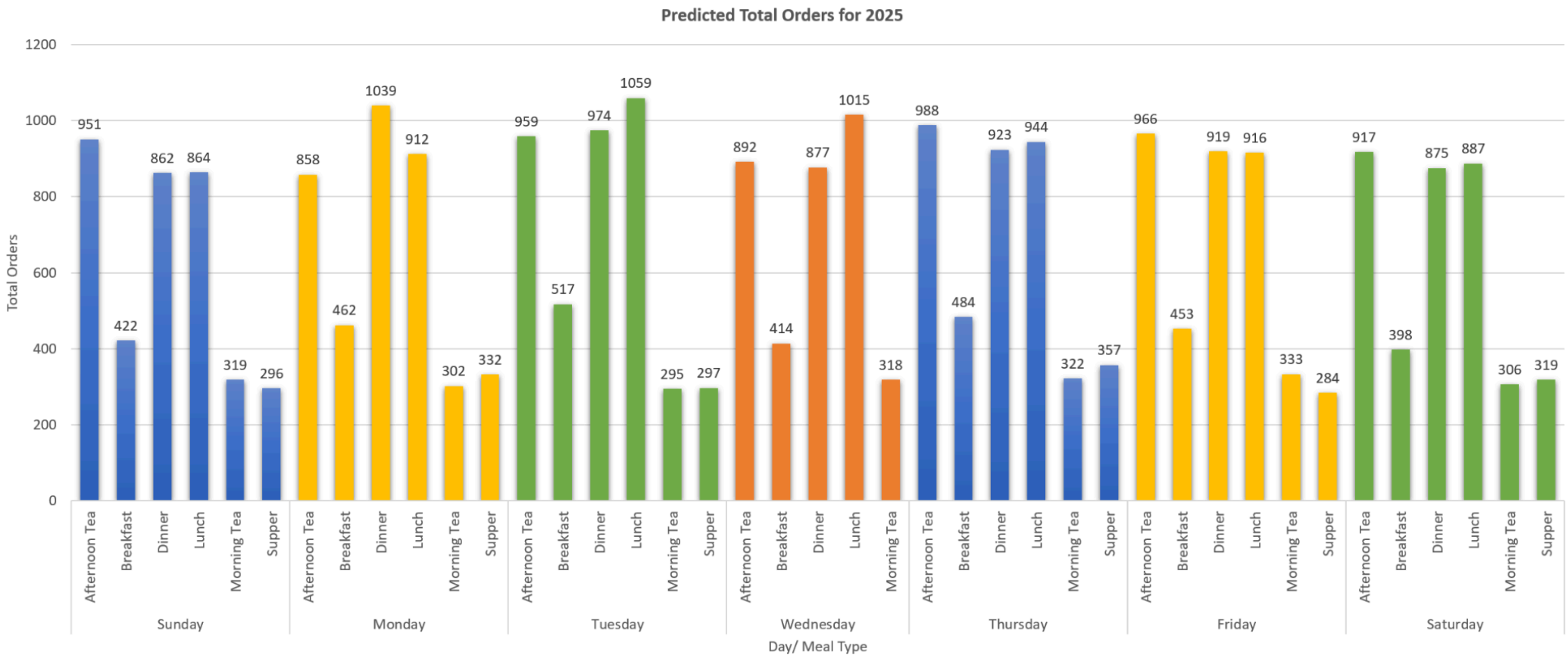
DAY	MEAL TYPE	2022 TOTAL ORDERS	2023 TOTAL ORDERS	2024 TOTAL ORDERS	AVERAGE ORDERS GROWTH %	2025 Predicted Orders	ORDERS DIFFERENCE
Friday	1)Breakfast	370	381	423	7	453	30
	2)Morning Tea	236	274	297	12	333	36
	3)Lunch	796	888	873	5	916	43
	4)Afternoon Tea	743	836	885	9	966	81
	5)Dinner	701	874	835	10	919	84
	6)Supper	273	293	280	1	284	4
**********		----------	----------	----------			
GRAND TOTAL:		3,119	3,546	3,593			
Saturday	1)Breakfast	385	368	393	1	398	5
	2)Morning Tea	248	269	285	7	306	21
	3)Lunch	734	859	830	7	887	57
	4)Afternoon Tea	771	861	865	6	917	52
	5)Dinner	742	871	825	6	875	50
	6)Supper	242	292	290	10	319	29
**********		----------	----------	----------			
GRAND TOTAL:		3,122	3,520	3,488			

Daily Orders by Meal Period Between The Year 2022 To 2024

Date: 21-SEP-24
Page No: 4

DAY	MEAL TYPE	2022 TOTAL ORDERS	2023 TOTAL ORDERS	2024 TOTAL ORDERS	AVERAGE ORDERS GROWTH %	2025 Predicted Orders	ORDERS DIFFERENCE
Sunday	1)Breakfast	350	411	395	7	422	27
	2)Morning Tea	239	286	289	10	319	30
	3)Lunch	713	837	808	7	864	56
	4)Afternoon Tea	688	871	849	12	951	102
	5)Dinner	789	867	836	3	862	26
	6)Supper	220	260	268	11	296	28
**********							
GRAND TOTAL:		2,999	3,532	3,445			

42 rows selected.



This graph shows the predicted total orders by meal type for each day in the year 2025.

This query is used to predict the daily orders by meal period for the year 2025. The value can be used by the managers to optimise staffing and kitchen operations during peak hours. For example, there are a predicted 1059 orders in the Tuesday Lunch Period which is 138 orders increases compared to 2024, the manager can arrange 7 more staff in this period to prevent delays in service, ensure faster order fulfilment, and maintain customer satisfaction. Also, for the Friday Supper period, there is a predicted 284 orders in total which is only 4 orders increases compared to 2024, the number of staff in this period can maintain the same as 2024, because the orders increase is not very significant.

3.3.2 Yearly Sales Performance by Each Stall Between The Year 2020 To 2024

SQL:

```
CL SCR

-- Set pagesize and linesize
SET LINESIZE 125
SET PAGESIZE 26

SELECT DISTINCT(stallname)
FROM menu_dim;

ACCEPT v_stall CHAR PROMPT 'Enter the Stall For Prediction : '

-- Format
COLUMN stallname                FORMAT A25              HEADING "STALL NAME"
COLUMN cal_year                 FORMAT '9999'           HEADING "YEAR"
COLUMN YearlyTotalSales         FORMAT '$99,999,999.99' HEADING "TOTAL    | SALES    "
COLUMN YearlyTotalOrder         FORMAT '99,999'         HEADING " TOTAL | ORDERS "
COLUMN SalesGrowthPerc          FORMAT '9999.9'         HEADING "SALES  | GROWTH %"
COLUMN OrderGrowthPerc          FORMAT '9999.9'         HEADING "ORDERS | GROWTH %"
COLUMN PredictedSales2025       FORMAT '$999,999.99'    HEADING "2025     | PREDICTED SALES"
COLUMN PredictedOrders2025      FORMAT '99,999'         HEADING "2025     | PREDICTED ORDERS"
COLUMN SalesPerOrder            FORMAT '$99,999.99'     HEADING "SALES  | PER ORDER"
BREAK ON stallname SKIP 1 ON cal_year

-- Title
```

```
TTITLE        CENTER 'Yearly Sales Performance And Prediction For ' &v_stall ' Between 2020 And 2024' -
        SKIP 1 -
        RIGHT 'Date: ' _DATE -
        SKIP 1 -
        RIGHT 'Page No: ' FORMAT 999 SQL.PNO -
        SKIP 2 -

COMPUTE AVG LABEL 'Average: '                OF SalesGrowthPerc  ON stallname
COMPUTE AVG                                  OF OrderGrowthPerc ON stallname
COMPUTE SUM LABEL 'GRAND TOTAL: '    OF YearlyTotalSales   ON stallname
COMPUTE SUM                                  OF YearlyTotalOrder  ON stallname

CREATE OR REPLACE VIEW StallYearlyView AS
WITH PreviousYear AS (
   SELECT
        M.stallname,
        D.cal_year,
        SUM(ITF.linetotal) AS YearlyTotalSales,
        COUNT(DISTINCT(ODF.orderid)) AS YearlyTotalOrder,
        LAG(SUM(ITF.linetotal)) OVER (PARTITION BY M.stallname ORDER BY D.cal_year) AS PrevYearSales,
        LAG(COUNT(DISTINCT(ODF.orderid))) OVER (PARTITION BY M.stallname ORDER BY D.cal_year) AS
PrevYearOrders
   FROM Orders_fact ODF
   JOIN Items_fact  ITF ON ODF.orderid = ITF.orderid
   JOIN Menu_dim    M   ON ITF.menu_key = M.menu_key
   JOIN Date_dim    D   ON ODF.date_key = D.date_key
   WHERE D.cal_year BETWEEN 2020 AND 2024
   GROUP BY M.stallname, D.cal_year
)
```

```
SELECT
	stallname,
	cal_year,
	YearlyTotalSales,
	YearlyTotalOrder,
	((YearlyTotalSales - PrevYearSales) / PrevYearSales) * 100 AS SalesGrowthPerc,
	((YearlyTotalOrder - PrevYearOrders) / PrevYearOrders) * 100 AS OrderGrowthPerc
FROM PreviousYear;

CREATE OR REPLACE VIEW PredictSales AS
SELECT
	SYV.stallname,
	SYV.cal_year,
	SYV.YearlyTotalSales,
	SYV.YearlyTotalOrder,
	SYV.SalesGrowthPerc,
	SYV.OrderGrowthPerc,
	CASE
		WHEN SYV.cal_year = 2024 THEN
			(SYV.YearlyTotalSales * (1 + (SELECT AVG(SalesGrowthPerc) FROM StallYearlyView WHERE stallname =
SYV.stallname) / 100))
	END AS PredictedSales2025,
	CASE
		WHEN SYV.cal_year = 2024 THEN
			(SYV.YearlyTotalOrder * (1 + (SELECT AVG(OrderGrowthPerc) FROM StallYearlyView WHERE stallname
= SYV.stallname) / 100))
	END AS PredictedOrders2025
FROM StallYearlyView SYV
ORDER BY SYV.stallname, SYV.cal_year;
```

```
SELECT
        PS.stallname,
        PS.cal_year,
        PS.YearlyTotalSales,
        PS.YearlyTotalOrder,
        PS.SalesGrowthPerc,
        PS.OrderGrowthPerc,
        PS.PredictedSales2025,
        PS.PredictedOrders2025,
        (PS.PredictedSales2025 / PS.PredictedOrders2025) AS SalesPerOrder
FROM PredictSales PS
WHERE PS.stallname = UPPER('&v_stall');

CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF
```

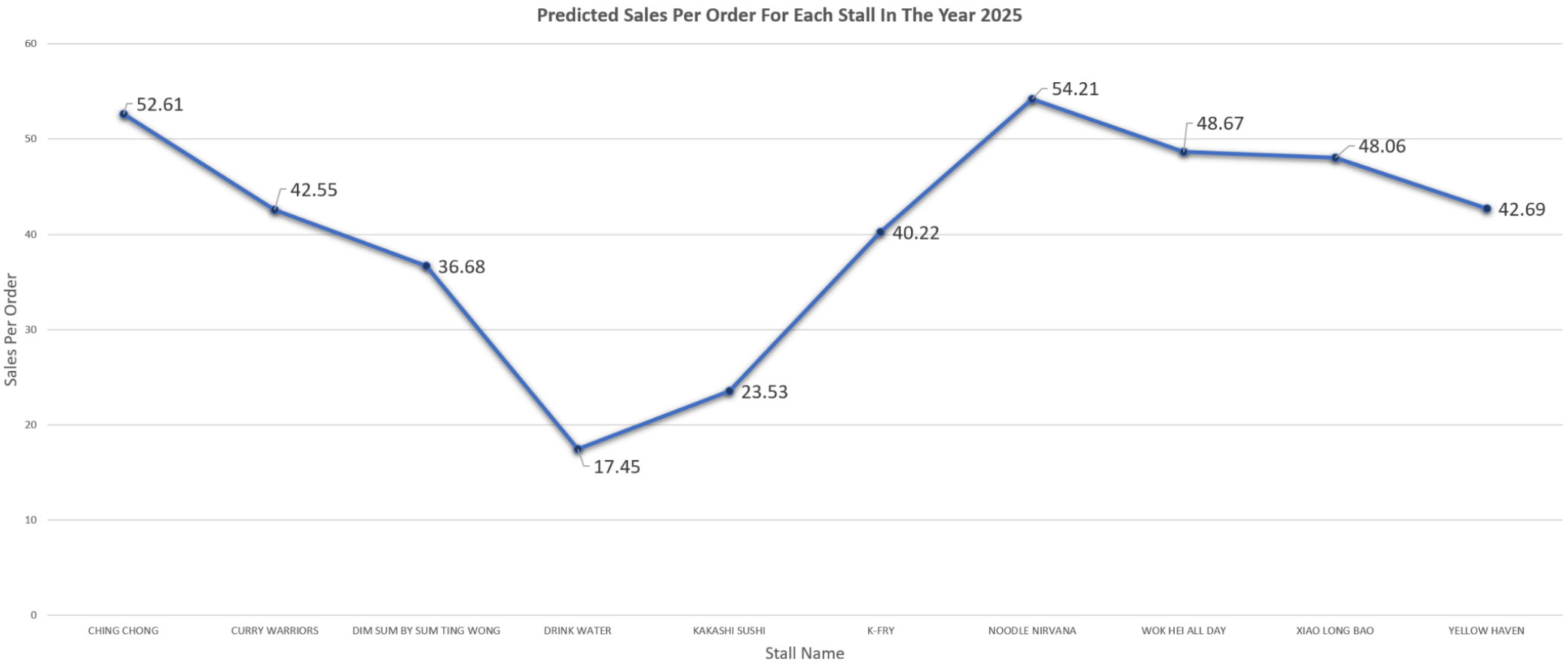
OUTPUT:

```
STALLNAME
-------------------------
XIAO LONG BAO
DRINK WATER
WOK HEI ALL DAY
DIM SUM BY SUM TING WONG
CURRY WARRIORS
KAKASHI SUSHI
CHING CHONG
YELLOW HAVEN
NOODLE NIRVANA
K-FRY

10 rows selected.

Enter the Stall For Prediction : DRINK WATER

View created.


View created.

old  11: WHERE PS.stallname = '&v_stall'
new  11: WHERE PS.stallname = 'DRINK WATER'
```

Yearly Sales Performance And Prediction For DRINKWATER Between 2020 And 2024

Date: 21-SEP-24
Page No: 1

STALL NAME	YEAR	TOTAL SALES	TOTAL ORDERS	SALES GROWTH %	ORDERS GROWTH %	2025 PREDICTED SALES	2025 PREDICTED ORDERS	SALES PER ORDER
DRINK WATER	2020	$139,966.00	5,620					
	2021	$140,985.00	5,596	.7	-.4			
	2022	$141,819.00	5,724	.6	2.3			
	2023	$86,293.50	4,301	-39.2	-24.9			
	2024	$62,677.50	3,381	-27.4	-21.4	$52,461.09	3,006	$17.45
**********	*****							
Average:				-16.3	-11.1			
GRAND TOTAL:		$571,741.00	24,622					



This graph shows the predicted sales per order for each stall in the year 2025.

The query is used to predict the sales per order for each stall in the year 2025. By analysing the predicted sales per order, the manager can assess the performance potential of each stall. Stalls with higher predicted sales per order indicate stronger performance and potential growth, while stalls with lower predicted values may be underperforming. This information can help the manager to replace the underperforming stall. For example, the stall DRINK WATER is contributing RM17.45 per order in the year 2025, it is the worst among all the other stalls, so the manager can replace or remove the stall DRINK WATER.

3.3.3 Stall Sales Performance Based On Customer Income Level Between The Year 2022 To 2024

SQL:

```
CL SCR

-- Set pagesize and linesize
SET LINESIZE 120
SET PAGESIZE 26

SELECT DISTINCT(stallname)
FROM menu_dim;

ACCEPT v_stall CHAR PROMPT 'Enter the Stall For Prediction : '

-- Format
COLUMN stallname                FORMAT A25              HEADING "STALL NAME"
COLUMN IncomeLevel              FORMAT A20              HEADING "      CUSTOMER |    INCOME LEVEL"
COLUMN TYearSales               FORMAT '$9,999,999.99'  HEADING "TOTAL   | SALES   "
COLUMN TYearOrders              FORMAT '99,999'         HEADING "TOTAL| ORDERS"
COLUMN AverageSales             FORMAT '$9,999,999.99'  HEADING "AVERAGE   |SALES PER YEAR"
COLUMN AverageOrders            FORMAT '99,999'         HEADING "AVERAGE   |ORDERS PER YEAR"
COLUMN AverageSalesPerOrder     FORMAT '$99,999.99'     HEADING "AVERAGE    | SALES PER ORDER"
BREAK ON stallname SKIP 1

-- Title
TTITLE          CENTER 'Stall Sales Performance Based On Customer Income Level Between 2022 And 2024' -
        SKIP 1 -
        RIGHT 'Date: ' _DATE -
        SKIP 1 -
        RIGHT 'Page No: ' FORMAT 999 SQL.PNO -
```

```
        SKIP 2 -

COMPUTE SUM LABEL 'GRAND TOTAL: '    OF TYearSales          ON stallname
COMPUTE SUM                          OF TYearOrders         ON stallname


CREATE OR REPLACE VIEW IncomeLevelView AS
SELECT
        customer_key,
        CASE
                WHEN MonthlyIncome = '2500 and below' OR MonthlyIncome = '2501 - 4000' THEN '3)LOW INCOME'
                WHEN MonthlyIncome = '4001 - 5500' OR MonthlyIncome = '5501 - 7000' THEN '2)MODERATE INCOME'
                ELSE '1)HIGH INCOME'
        END AS IncomeLevel
FROM customer_dim;


CREATE OR REPLACE VIEW ThreeYearSales AS
SELECT
        M.stallname,
        ILV.IncomeLevel,
        SUM(ITF.linetotal) AS TYearSales
FROM orders_fact            ODF
JOIN items_fact             ITF     ON ODF.orderid = ITF.orderid
JOIN date_dim               D       ON ODF.date_key = D.date_key
JOIN IncomeLevelView        ILV     ON ODF.customer_key = ILV.customer_key
JOIN menu_dim               M       ON M.menu_key = ITF.menu_key
WHERE D.cal_year BETWEEN 2022 AND 2024
GROUP BY M.stallname, ILV.IncomeLevel;
```

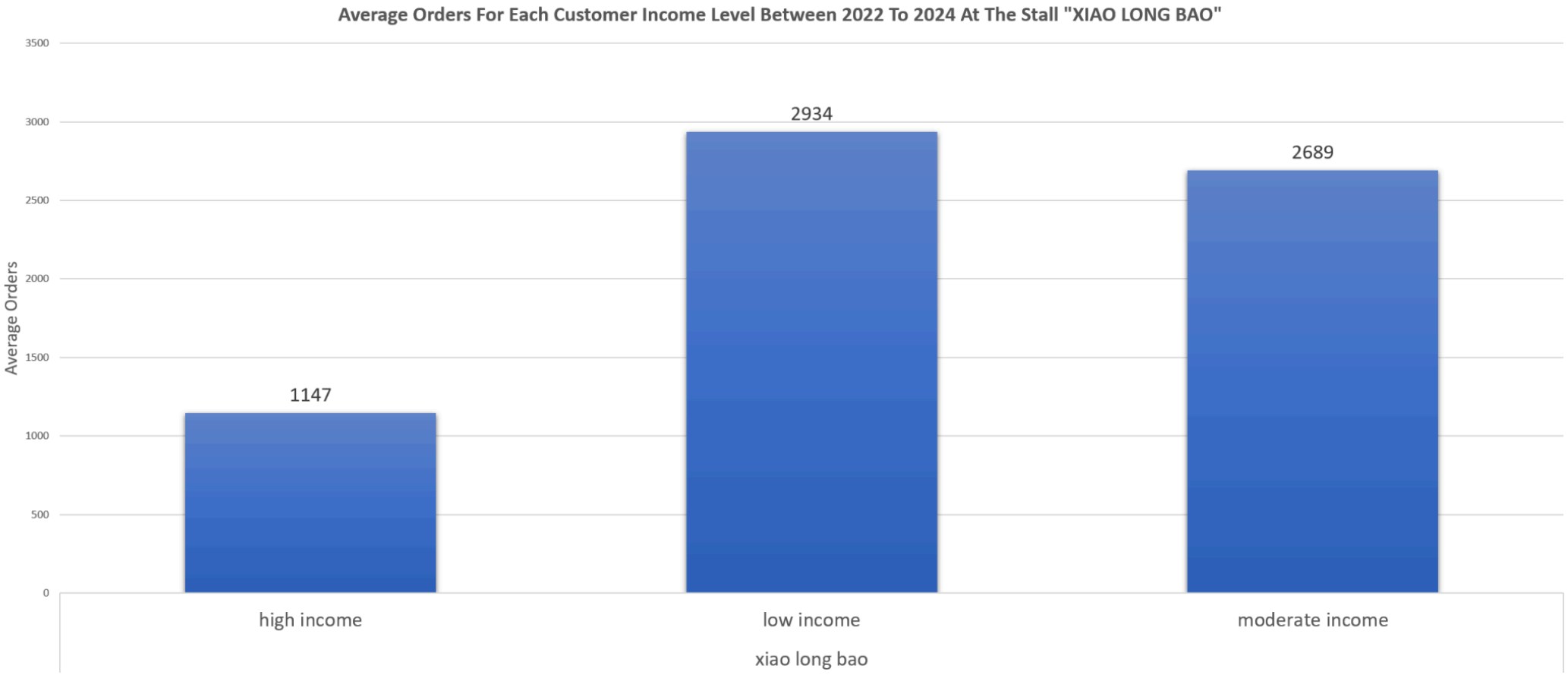
```
CREATE OR REPLACE VIEW ThreeYearOrders AS
SELECT
        M.stallname,
        ILV.IncomeLevel,
        COUNT(DISTINCT(ODF.orderid)) AS TYearOrders
FROM orders_fact        ODF
JOIN items_fact         ITF     ON ODF.orderid = ITF.orderid
JOIN date_dim           D       ON ODF.date_key = D.date_key
JOIN IncomeLevelView    ILV     ON ODF.customer_key = ILV.customer_key
JOIN menu_dim           M       ON M.menu_key = ITF.menu_key
WHERE D.cal_year BETWEEN 2022 AND 2024
GROUP BY M.stallname, ILV.IncomeLevel;

CREATE OR REPLACE VIEW AverageSalesView AS
SELECT
        M.stallname,
        ILV.IncomeLevel,
        (SUM(ITF.linetotal) / 3) AS AverageSales
FROM orders_fact        ODF
JOIN items_fact         ITF     ON ODF.orderid = ITF.orderid
JOIN date_dim           D       ON ODF.date_key = D.date_key
JOIN IncomeLevelView    ILV     ON ODF.customer_key = ILV.customer_key
JOIN menu_dim           M       ON M.menu_key = ITF.menu_key
WHERE D.cal_year BETWEEN 2022 AND 2024
GROUP BY M.stallname, ILV.IncomeLevel;

CREATE OR REPLACE VIEW AverageOrdersView AS
SELECT
```

```
        M.stallname,
        ILV.IncomeLevel,
        (COUNT(DISTINCT(ODF.orderid)) / 3) AS AverageOrders
FROM orders_fact        ODF
JOIN items_fact         ITF     ON ODF.orderid = ITF.orderid
JOIN date_dim           D       ON ODF.date_key = D.date_key
JOIN IncomeLevelView    ILV     ON ODF.customer_key = ILV.customer_key
JOIN menu_dim           M       ON M.menu_key = ITF.menu_key
WHERE D.cal_year BETWEEN 2022 AND 2024
GROUP BY M.stallname, ILV.IncomeLevel;


CREATE OR REPLACE VIEW CompareView AS
SELECT
        TYS.stallname,
        TYS.IncomeLevel,
        TYS.TYearSales,
        ASV.AverageSales,
        TYO.TYearOrders,
        AOV.AverageOrders
FROM ThreeYearSales     TYS
JOIN AverageSalesView   ASV   ON TYS.stallname = ASV.stallname AND TYS.IncomeLevel = ASV.IncomeLevel
JOIN ThreeYearOrders    TYO   ON TYS.stallname = TYO.stallname AND TYS.IncomeLevel = TYO.IncomeLevel
JOIN AverageOrdersView  AOV   ON TYS.stallname = AOV.stallname AND TYS.IncomeLevel = AOV.IncomeLevel;

SELECT
        stallname,
        IncomeLevel,
        TYearSales,
```

```
        TYearOrders,
        AverageSales,
        AverageOrders,
        (AverageSales / AverageOrders) AS AverageSalesPerOrder
FROM CompareView CV
WHERE stallname = UPPER('&v_stall')
ORDER BY stallname, IncomeLevel;

CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF
```

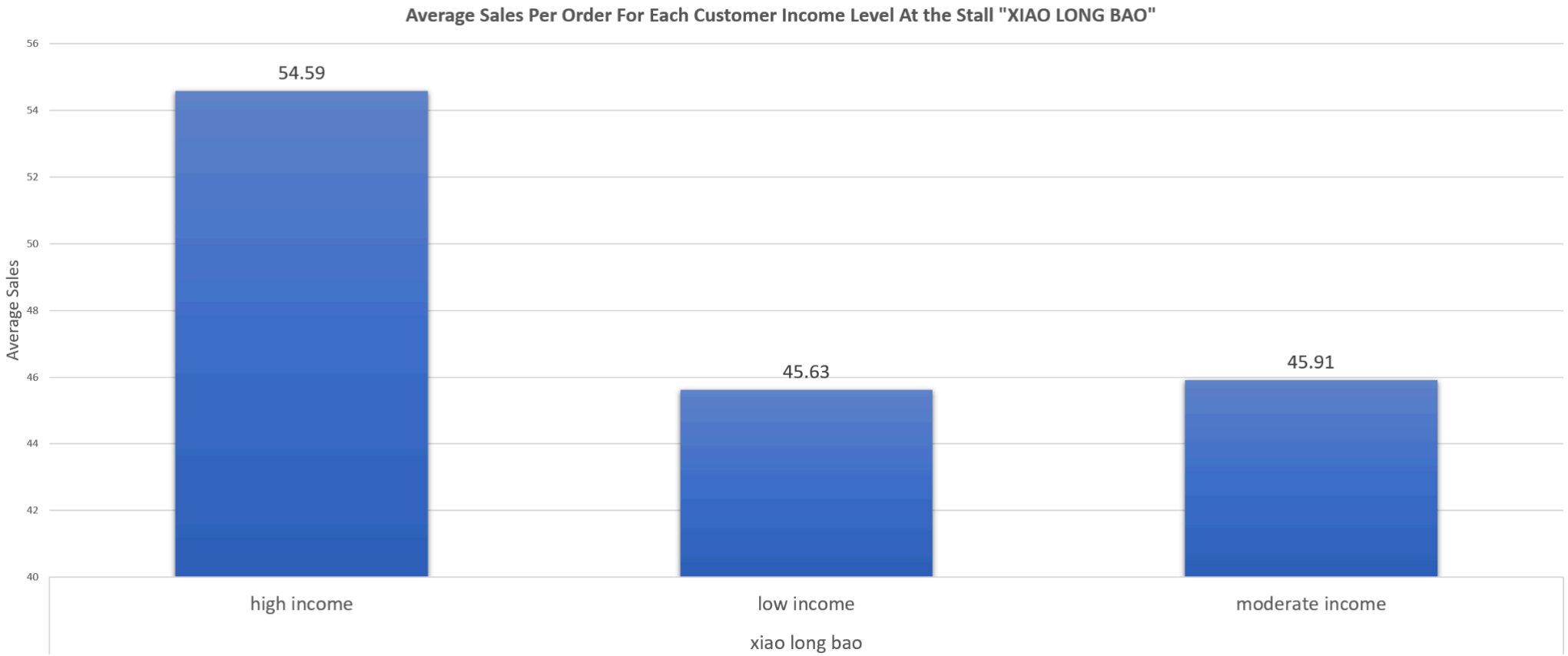
OUTPUT:

```
Enter the Stall For Analysis : xiao long bao

old   9: WHERE stallname = UPPER('&v_stall')
new   9: WHERE stallname = UPPER('xiao long bao')

                    Stall Sales Performance Based On Customer Income Level Between 2022 And 2024
                                                                                    Date: 22-SEP-24
                                                                                      Page No:    1

                              CUSTOMER             TOTAL      TOTAL     AVERAGE         AVERAGE         AVERAGE
STALL NAME                 INCOME LEVEL            SALES     ORDERS SALES PER YEAR ORDERS PER YEAR  SALES PER ORDER
------------------------- -------------------- ------------- ------- ------------- --------------- ----------------
XIAO LONG BAO             1)HIGH INCOME          $187,849.50   3,441     $62,616.50           1,147           $54.59
                          2)MODERATE INCOME      $370,377.50   8,067    $123,459.17           2,689           $45.91
                          3)LOW INCOME           $401,569.00   8,801    $133,856.33           2,934           $45.63
************************                       ------------- -------
GRAND TOTAL:                                     $959,796.00  20,309
```



This graph shows the average orders for each customer income level between the year 2022 to 2024 at the stall “XIAO LONG BAO”.

This query is used to analyse the stall performance based on customer income level. As the graph above shows that only 1147 of high income customers order at the stall “XIAO LONG BAO”, which is the lowest among other customer income levels. The manager of the stall needs to introduce a new meal bundle to attract the high income customers to order at the “XIAO LONG BAO” stall.



This graph shows the average sales per order for each customer income level at the stall “XIAO LONG BAO”.

As the graph above shows that the average spending of high income customers per order is RM54.59, so the manager can utilise this information to introduce the new meal bundle with the price around RM 54.59 to attract more high income customers to spend on the stall.

3.4 Ng Hong Han

3.4.1 Analysing Annual Sales Performance by Period

SQL:

```
-- View 1: Avg_Last_Week_Month2
CREATE OR REPLACE VIEW Avg_Last_Week_Month2 AS
WITH Last_Week_Periods AS (
    SELECT
        D.cal_year,
        D.cal_year_month,
        SUM(o.orderAmount) AS total_sales,  -- Sum total_sales instead of individual orderAmount
        COUNT(o.orderID) AS order_count
    FROM
        Orders_Fact o
    JOIN
        date_dim D ON o.date_key = D.date_key
    WHERE
        D.cal_date >= (SELECT MAX(cal_date) - 6
                       FROM date_dim D2
                       WHERE D2.cal_year_month = D.cal_year_month)
        AND D.cal_date <= (SELECT MAX(cal_date)
                           FROM date_dim D2
                           WHERE D2.cal_year_month = D.cal_year_month)
    GROUP BY
        D.cal_year, D.cal_year_month
),
Aggregated_Sales AS (
    SELECT
        cal_year,
        SUM(total_sales) AS total_sales,
        SUM(order_count) AS total_orders
    FROM
```

```
        Last_Week_Periods
    GROUP BY
        cal_year
)
SELECT
    cal_year AS Year,
    total_sales,
    total_orders, -- Keep total_orders if needed for reporting
    total_sales / 12 AS avg_last_week_sales -- Corrected calculation
FROM
    Aggregated_Sales
ORDER BY
    cal_year;


-- View 2: Avg_Specific_Days2
CREATE OR REPLACE VIEW Avg_Specific_Days2 AS
WITH Specific_Days AS (
    SELECT
        cal_year,
        COUNT(DISTINCT cal_month_year) AS number_of_months
    FROM
        date_dim
    WHERE
        day_number_month = TO_NUMBER(SUBSTR(cal_year_month, 6, 2))
    GROUP BY
        cal_year
),
Sales_Specific_Days AS (
    SELECT
        D.cal_year,
        D.cal_month_year,
        SUM(o.orderAmount) AS total_sales -- Sum total_sales for accuracy
    FROM
```

```
        Orders_Fact o
    JOIN
        date_dim D ON o.date_key = D.date_key
    WHERE
        D.day_number_month = TO_NUMBER(SUBSTR(D.cal_year_month, 6, 2))
    GROUP BY
        D.cal_year, D.cal_month_year
),
Aggregated_Sales AS (
    SELECT
        cal_year,
        SUM(total_sales) AS total_sales
    FROM
        Sales_Specific_Days
    GROUP BY
        cal_year
)
SELECT
    a.cal_year AS Year,
    a.total_sales,
    a.total_sales / n.number_of_months AS avg_sales_specific_days
FROM
    Aggregated_Sales a
JOIN
    Specific_Days n ON a.cal_year = n.cal_year
ORDER BY
    a.cal_year;


-- View 3: Avg_Remaining_Days2
CREATE OR REPLACE VIEW Avg_Remaining_Days2 AS
WITH Remaining_Days_Sales AS (
    SELECT
        D.cal_year,
```

```
        SUM(o.orderAmount) AS total_remaining_days_sales,
        COUNT(o.orderID) AS total_orders,
        CASE
            WHEN MOD(D.cal_year, 4) = 0 AND (MOD(D.cal_year, 100) != 0 OR MOD(D.cal_year, 400) = 0)
            THEN 366
            ELSE 365
        END AS total_days_in_year
    FROM
        Orders_Fact o
    JOIN
        date_dim D ON o.date_key = D.date_key
    WHERE
        D.cal_date NOT IN (
            SELECT cal_date
            FROM date_dim D2
            WHERE D2.cal_date >= (SELECT MAX(cal_date) - 6
                                  FROM date_dim D3
                                  WHERE D3.cal_year_month = D2.cal_year_month)
            AND D2.cal_date <= (SELECT MAX(cal_date)
                                FROM date_dim D3
                                WHERE D3.cal_year_month = D2.cal_year_month)
        )
        AND D.day_number_month != TO_NUMBER(SUBSTR(D.cal_year_month, 6, 2))
    GROUP BY
        D.cal_year
)
SELECT
    cal_year AS Year,
    total_remaining_days_sales,
    -- Calculate the remaining days in year (based on total days)
    total_days_in_year - (SELECT COUNT(DISTINCT cal_date)
                          FROM date_dim
                          WHERE cal_year = Remaining_Days_Sales.cal_year
                          AND day_number_month = TO_NUMBER(SUBSTR(cal_year_month, 6, 2))
```

```
                            AND cal_date < (SELECT MAX(cal_date)
                                            FROM date_dim
                                            WHERE cal_year = Remaining_Days_Sales.cal_year)) - 7 AS
remaining_days_in_year,
    total_remaining_days_sales / 12 AS avg_remaining_days_sales -- Average calculation updated
FROM
    Remaining_Days_Sales
ORDER BY
    Year;


-- Final Report Query
SET LINESIZE 170
SET PAGESIZE 50

TTITLE CENTER 'Annual Average Sales Report from 2020 - 2024: Last Week vs. Remaining Days' SKIP 1 -
LEFT 'Date Generated: ' _DATE -
RIGHT 'Page: ' FORMAT 999 SQL.PNO SKIP 2
BREAK ON REPORT

COLUMN Year FORMAT 9999 HEADING "Years";
COLUMN total_last_week_sales FORMAT RM9,999,999.00 HEADING "Total Last Week of Month Sales";
COLUMN total_remaining_days_sales FORMAT RM99,999,999.00 HEADING "Total Remaining Days of Month
Sales";
COLUMN avg_last_week_sales FORMAT RM999,999.00 HEADING "Average Last Week of Month Sales";
COLUMN avg_remaining_days_sales FORMAT RM999,999.00 HEADING "Average Remaining Days of Month Sales";

COMPUTE SUM LABEL 'Total: ' OF total_last_week_sales ON REPORT
COMPUTE SUM LABEL 'Total: ' OF total_remaining_days_sales ON REPORT
COMPUTE AVG LABEL 'Aver: ' OF avg_last_week_sales ON REPORT
COMPUTE AVG LABEL 'Aver: ' OF avg_remaining_days_sales ON REPORT;

SELECT
    A.Year,
```

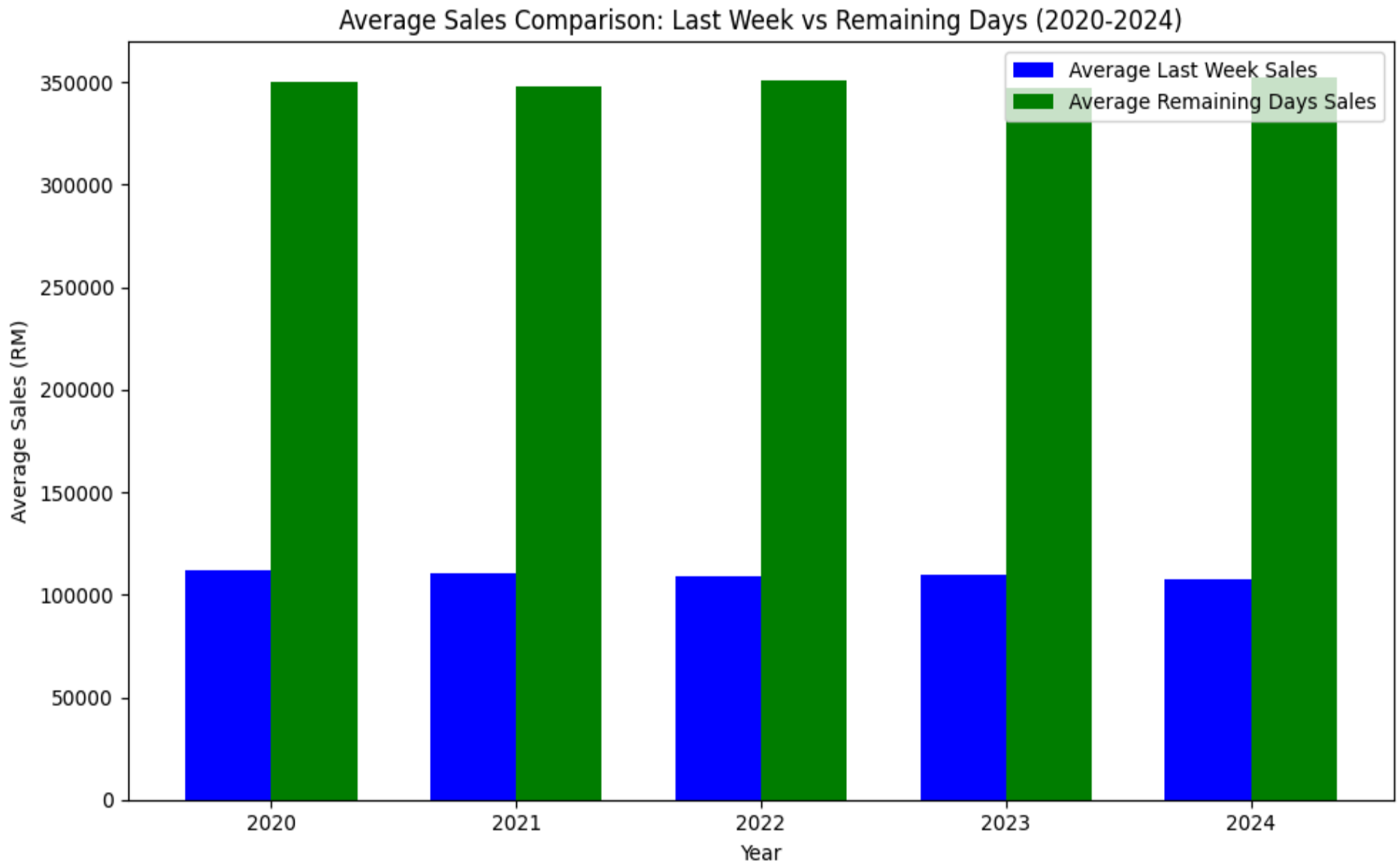
```
    A.total_sales AS total_last_week_sales, -- Total sales for the last week of the month
    C.total_remaining_days_sales, -- Total sales for the remaining days of the month
    A.avg_last_week_sales,
    C.avg_remaining_days_sales
FROM
    Avg_Last_Week_Month2 A
JOIN
    Avg_Specific_Days2 B ON A.Year = B.Year
JOIN
    Avg_Remaining_Days2 C ON B.Year = C.Year
WHERE
    A.Year BETWEEN 2020 AND 2024 -- Filter to show only 2020 to 2024
ORDER BY
    A.Year;
```

OUTPUT:

Annual Average Sales Report from 2020 - 2024: Last Week vs. Remaining Days

Date Generated: 22-SEP-24
Page: 1

Years	Total Last Week of Month Sales	Total Remaining Days of Month Sales	Average Last Week of Month Sales	Average Remaining Days of Month Sales
2020	$1,341,001.50	$4,203,664.00	$111,750.13	$350,305.33
2021	$1,323,011.00	$4,175,018.00	$110,250.92	$347,918.17
2022	$1,307,369.00	$4,215,029.00	$108,947.42	$351,252.42
2023	$1,317,901.50	$4,165,306.00	$109,825.13	$347,108.83
2024	$1,291,195.50	$4,225,464.00	$107,599.63	$352,122.00
Aver:			$109,674.64	$349,741.35
Total	$6,580,478.50	$20,984,481.00		


Average Sales Comparison: Last Week vs Remaining Days (2020-2024)
Average Last Week Sales
Average Remaining Days Sales
Average Sales (RM)
350000
300000
250000
200000
150000
100000
50000
0
2020
2021
2022
2023
2024
Year

This query generates an Annual Average Sales Report that compares sales from the last week of each month to the remaining days for the years 2020 to 2024. It reveals a declining trend in last week sales, decreasing from RM1,341,001.50 in 2020 to RM1,291,195.50 in 2024, while sales during the remaining days remain consistently higher, with average daily sales of $349,741.35 compared to RM109,674.64 for the last week. This suggests potential inefficiencies or missed opportunities during the final week. To improve last-week sales performance, businesses can implement several targeted strategies. First, launching specific marketing campaigns like flash sales, limited-time discounts, or end-of-month offers can create urgency and drive purchases. Offering customer incentives, such as double loyalty points or personalised last-week discounts, can further encourage buying during this period. Operational improvements are also important, ensuring that popular items are fully stocked and optimising staffing levels can enhance the customer experience. Gathering customer feedback through surveys can provide insights into why sales drop, allowing the business to tailor promotions based on consumer behaviour.

3.4.2 Top 5 Sales Based On State At Each Stall In the Year 2023 and 2024

SQL:

```
-- Set pagesize and linesize
SET LINESIZE 120
SET PAGESIZE 250

-- Format
COLUMN stallname         FORMAT A25            HEADING "STALL NAME"
COLUMN state             FORMAT A20            HEADING "STATE"
COLUMN TOTALSALES1       FORMAT '$9999999999.99'    HEADING "2024   | Sales   "
COLUMN TOTALSALES2       FORMAT '$9999999999.99'    HEADING "2023   | Sales   "
COLUMN SalesDifference   FORMAT '$9999999999.99'    HEADING "SALES DIFFERENCE"
BREAK ON stallname SKIP 2

-- Title
TTITLE    CENTER 'Top 5 Sales Based On State At Each Stall In the Year 2023 And 2024' -
          SKIP 1 -
          RIGHT 'Date: ' _DATE -
          SKIP 1 -
          RIGHT 'Page No: ' FORMAT 999 SQL.PNO -
          SKIP 2 -

CREATE OR REPLACE VIEW Sales1 AS
SELECT M.stallname, C.state, SUM(ITF.linetotal) AS TotalSales1
FROM orders_fact      ODF
JOIN items_fact       ITF  ON ODF.orderid = ITF.orderid
JOIN date_dim         D    ON ODF.date_key = ITF.date_key
JOIN menu_dim         M    ON ITF.menu_key = M.menu_key
JOIN customer_dim     C    ON ODF.customer_key = C.customer_key
WHERE D.cal_year = '2024'
GROUP BY M.stallname, C.state;
```

```
CREATE OR REPLACE VIEW Sales2 AS
SELECT M.stallname, C.state, SUM(ITF.linetotal) AS TotalSales2
FROM orders_fact      ODF
JOIN items_fact       ITF  ON ODF.orderid = ITF.orderid
JOIN date_dim         D    ON ODF.date_key = ITF.date_key
JOIN menu_dim         M    ON ITF.menu_key = M.menu_key
JOIN customer_dim     C    ON ODF.customer_key = C.customer_key
WHERE D.cal_year = '2023'
GROUP BY M.stallname, C.state;

CREATE OR REPLACE VIEW CompareView AS
SELECT S1.stallname, S1.state, TotalSales1, TotalSales2, (TotalSales1 - TotalSales2) AS
SalesDifference
FROM Sales1  S1
JOIN Sales2  S2 ON S1.stallname = S2.stallname AND S1.state = S2.state;

-- Now get only the top 5 based on TotalSales1 and TotalSales2 for each stall
WITH RankedSales AS (
  SELECT
    stallname,
    state,
    TotalSales1,
    TotalSales2,
    SalesDifference,
    ROW_NUMBER() OVER (PARTITION BY stallname ORDER BY TotalSales1 DESC) AS rank2023,
    ROW_NUMBER() OVER (PARTITION BY stallname ORDER BY TotalSales2 DESC) AS rank2024
  FROM CompareView
)
SELECT stallname, state, TotalSales2, TotalSales1, SalesDifference
FROM RankedSales
WHERE rank2023 <= 5 OR rank2024 <= 5
ORDER BY stallname, TotalSales1 DESC;
```

```
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF;
```

OUTPUT:

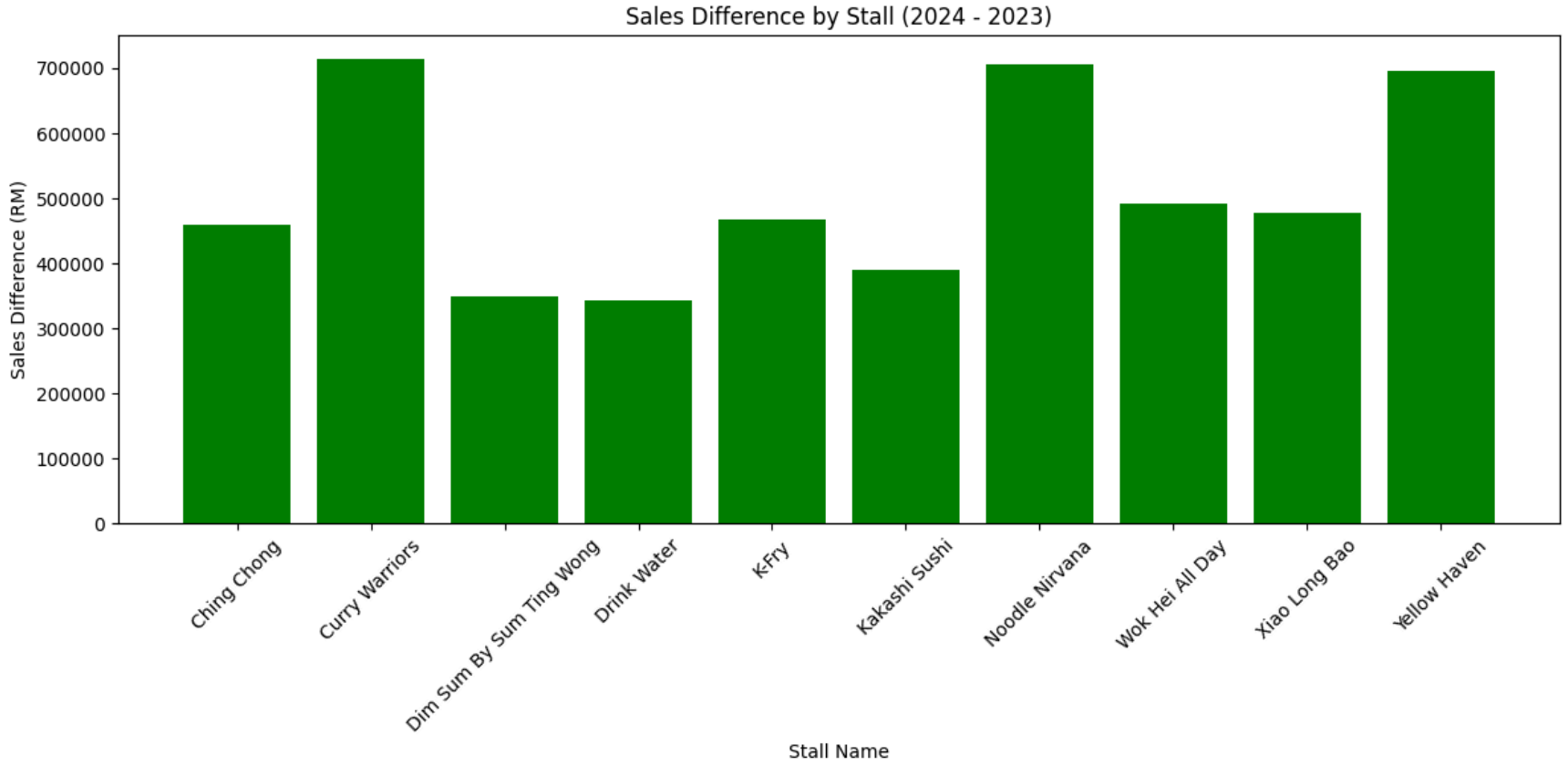
Top 5 Sales Based On State At Each Stall In the Year 2023 And 2024

Date: 22-SEP-24
Page No: 1

STALL NAME	STATE	2023 Sales	2024 Sales	SALES DIFFERENCE
CHING CHONG	PERLIS	$167600882.50	$168060063.00	$459180.50
	KUALA LUMPUR	$146454425.00	$146855670.00	$401245.00
	PAHANG	$145694312.50	$146093475.00	$399162.50
	SABAH	$144626140.00	$145022376.00	$396236.00
	PENANG	$129767720.00	$130123248.00	$355528.00
CURRY WARRIORS	PERLIS	$260776805.00	$261491262.00	$714457.00
	KUALA LUMPUR	$240337170.00	$240995628.00	$658458.00
	SABAH	$232426707.50	$233063493.00	$636785.50
	PAHANG	$231917715.00	$232553106.00	$635391.00
	PUTRAJAYA	$208135592.50	$208705827.00	$570234.50
DIM SUM BY SUM TING WONG	PERLIS	$127785770.00	$128135868.00	$350098.00
	KUALA LUMPUR	$116413465.00	$116732406.00	$318941.00
	PAHANG	$111383400.00	$111688560.00	$305160.00
	SABAH	$110427830.00	$110730372.00	$302542.00
	PENANG	$97678745.00	$97946358.00	$267613.00
DRINK WATER	PERLIS	$124926177.50	$125268441.00	$342263.50
	KUALA LUMPUR	$114345010.00	$114658284.00	$313274.00
	PAHANG	$113434882.50	$113745663.00	$310780.50
	SABAH	$110990842.50	$111294927.00	$304084.50
	PUTRAJAYA	$99652300.00	$99925320.00	$273020.00
K-FRY	PERLIS	$170745905.00	$171213702.00	$467797.00
	PAHANG	$154520012.50	$154943355.00	$423342.50
	SABAH	$153785450.00	$154206780.00	$421330.00
	KUALA LUMPUR	$153355115.00	$153775266.00	$420151.00
	MALACCA	$139914902.50	$140298231.00	$383328.50
KAKASHI SUSHI	PERLIS	$142044495.00	$142433658.00	$389163.00

	PAHANG	$125017975.00	$125360490.00	$342515.00
	SABAH	$124819050.00	$125161020.00	$341970.00
	KUALA LUMPUR	$122415890.00	$122751276.00	$335386.00
	PENANG	$110381840.00	$110684256.00	$302416.00
NOODLE NIRVANA	PERLIS	$257932360.00	$258639024.00	$706664.00
	KUALA LUMPUR	$238128555.00	$238780962.00	$652407.00
	SABAH	$236150985.00	$236797974.00	$646989.00
	PAHANG	$229197735.00	$229825674.00	$627939.00
	MALACCA	$209184420.00	$209757528.00	$573108.00
WOK HEI ALL DAY	PERLIS	$179653730.00	$180145932.00	$492202.00
	PAHANG	$163953255.00	$164402442.00	$449187.00
	SABAH	$160810240.00	$161250816.00	$440576.00
	KUALA LUMPUR	$160157255.00	$160596042.00	$438787.00
	MALACCA	$144664830.00	$145061172.00	$396342.00
XIAO LONG BAO	PERLIS	$174610160.00	$175088544.00	$478384.00
	KUALA LUMPUR	$160063632.50	$160502163.00	$438530.50
	SABAH	$158713132.50	$159147963.00	$434830.50
	PAHANG	$158228777.50	$158662281.00	$433503.50
	PENANG	$142840012.50	$143231355.00	$391342.50
YELLOW HAVEN	PERLIS	$253812970.00	$254508348.00	$695378.00
	KUALA LUMPUR	$227142967.50	$227765277.00	$622309.50
	PAHANG	$225526382.50	$226144263.00	$617880.50
	SABAH	$222047385.00	$222655734.00	$608349.00
	MALACCA	$199405522.50	$199951839.00	$546316.50

50 rows selected.


Sales Difference by Stall (2024 - 2023)
Sales Difference (RM)
0
100000
200000
300000
400000
500000
600000
700000
Ching Chong
Curry Warriors
Dim Sum By Sum Ting Wong
Drink Water
K-Fry
Kakashi Sushi
Noodle Nirvana
Wok Hei All Day
Xiao Long Bao
Yellow Haven
Stall Name

This query summarises the sales performance of various stalls across different states in 2023 and 2024, highlighting the top five spending customers for each stall along with sales figures and differences year-over-year. It allows for the identification of trends, revealing geographic strengths and weaknesses, while the "Sales Difference" column indicates overall market growth or decline. This data provides valuable market insights, helping to pinpoint high-performing areas where marketing efforts can be intensified, as well as underperforming stalls that may require further analysis to understand the causes behind their stagnation. High-performing regions, like Perlis or Kuala Lumpur, should receive increased marketing efforts, loyalty programs, and events to attract customer engagement, while underperforming stalls require investigation into root causes, such as competition or product alignment, to develop corrective strategies. Targeted promotional campaigns can boost sales in weaker regions through discounts, bundles, or localised marketing tailored to customer preferences.

3.4.3 Top 5 Items Ordered by Income Group

SQL:

```
SET PAGESIZE 50;
SET LINESIZE 110;
ALTER SESSION SET NLS_DATE_FORMAT = 'dd/MM/YYYY';

-- Title setup
TTITLE CENTER 'Top 5 Items Ordered by Income Group' -
    SKIP 1 -
    RIGHT 'Date: ' _DATE -
    SKIP 1 -
    RIGHT 'Page: ' FORMAT 999 SQL.PNO -
    SKIP 2;

-- Formatting for output
COLUMN income_group FORMAT A20 HEADING "Income | Group";
COLUMN item_name FORMAT A30 HEADING "Item Name";
COLUMN order_count FORMAT 9999 HEADING "Total | Orders";
COLUMN total_sales FORMAT 9,999,999.99 HEADING "Total | Sales (RM)";

BREAK ON income_group SKIP 1;

WITH customer_spending AS (
    SELECT
        cd.monthlyIncome AS income_group,
        m.itemName,
        COUNT(*) AS order_count,
        SUM(ofact.orderAmount) AS total_sales  -- Calculate total sales for each item
    FROM Orders_Fact ofact
    JOIN customer_dim cd ON ofact.customer_key = cd.customer_key
    JOIN Items_Fact i ON ofact.orderID = i.orderID
    JOIN menu_dim m ON i.menu_key = m.menu_key
    GROUP BY cd.monthlyIncome, m.itemName
```

```
),
ranked_items AS (
    SELECT
        income_group,
        itemName,
        order_count,
        total_sales,
        ROW_NUMBER() OVER (PARTITION BY income_group ORDER BY order_count DESC) AS rank
    FROM customer_spending
)
SELECT
    income_group,
    itemName,
    order_count,
    total_sales
FROM ranked_items
WHERE rank <= 5  -- Limit to top 5 items per income group
ORDER BY income_group, order_count DESC;

-- Clear any formatting changes after the report
TTITLE OFF;
SET VERIFY OFF;
```

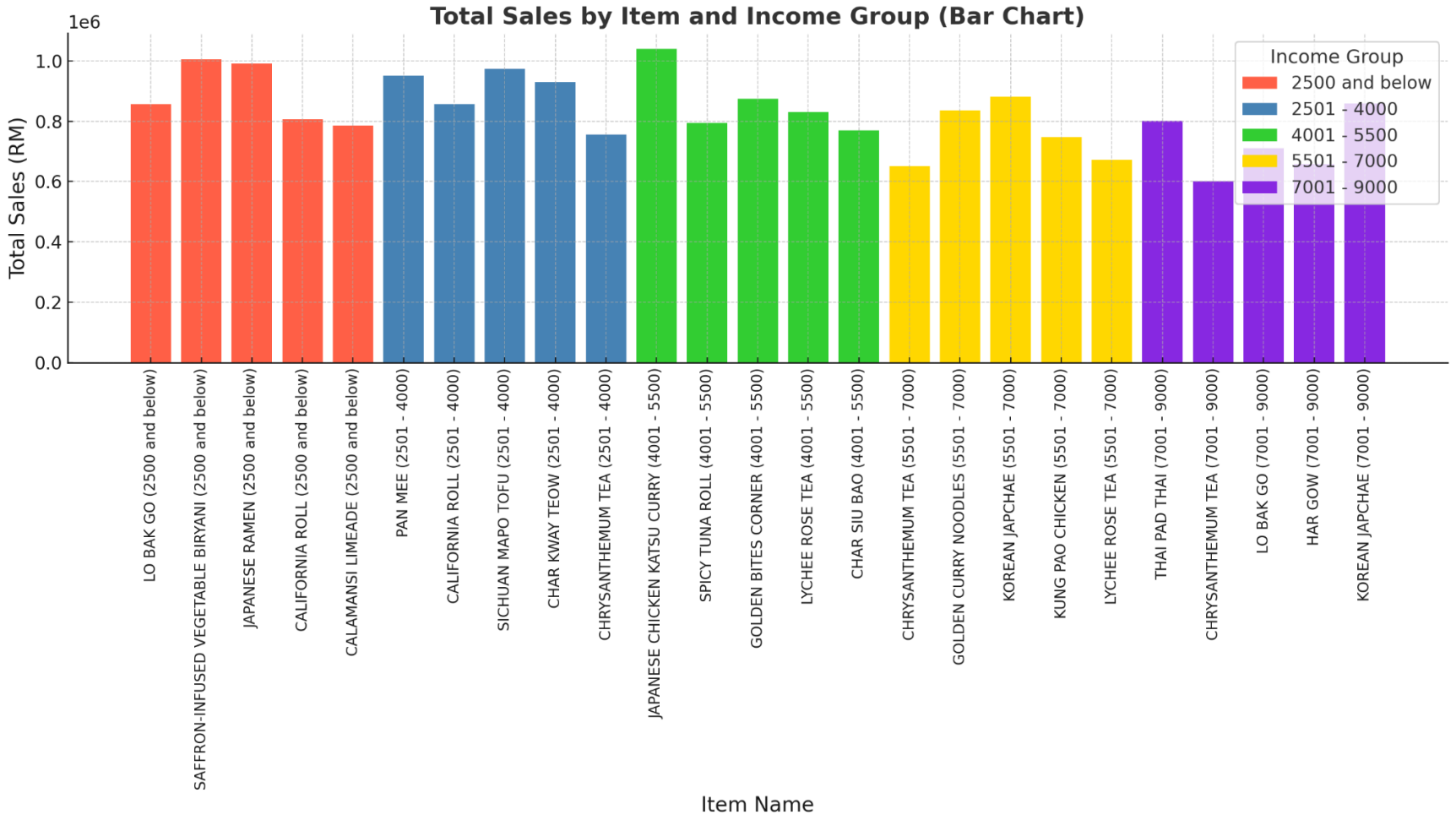
OUTPUT:

Top 5 Items Ordered by Income Group

Date:
Page: 1

Income Group	ITEMNAME	Total Orders	Total Sales (RM)
2500 and below	LO BAK GO	2585	856,703.50
	SAFFRON-INFUSED VEGETABLE BIRYANI	2528	1,004,358.00
	JAPANESE RAMEN	2525	990,211.50
	CALIFORNIA ROLL	2514	806,267.00
	CALAMANSI LIMEADE	2513	785,609.00
2501 - 4000	PAN MEE	2747	951,058.00
	CALIFORNIA ROLL	2733	856,741.00
	SICHUAN MAPO TOFU	2678	974,219.50
	CHAR KWAY TEOW	2674	929,150.50
	CHRYSANTHEMUM TEA	2670	756,288.50
4001 - 5500	JAPANESE CHICKEN KATSU CURRY	2651	1,040,333.00
	SPICY TUNA ROLL	2649	793,415.50
	GOLDEN BITES CORNER	2647	873,698.50
	LYCHEE ROSE TEA	2634	830,798.50
	CHAR SIU BAO	2629	770,055.00
5501 - 7000	CHRYSANTHEMUM TEA	2225	650,607.00
	GOLDEN CURRY NOODLES	2218	835,515.00
	KOREAN JAPCHAE	2217	880,320.00
	KUNG PAO CHICKEN	2206	746,657.50
	LYCHEE ROSE TEA	2199	672,504.00
7001 - 9000	THAI PAD THAI	2091	801,002.00
	CHRYSANTHEMUM TEA	2067	601,250.00
	LO BAK GO	2062	709,926.50
	HAR GOW	2057	658,881.00
	KOREAN JAPCHAE	2052	859,036.50

25 rows selected.


Total Sales by Item and Income Group (Bar Chart)
1e6
Total Sales (RM)
0.0
0.2
0.4
0.6
0.8
1.0
Income Group
2500 and below
2501 - 4000
4001 - 5500
5501 - 7000
7001 - 9000
LO BAK GO (2500 and below)
SAFFRON-INFUSED VEGETABLE BIRYANI (2500 and below)
JAPANESE RAMEN (2500 and below)
CALIFORNIA ROLL (2500 and below)
CALAMANSI LIMEADE (2500 and below)
PAN MEE (2501 - 4000)
CALIFORNIA ROLL (2501 - 4000)
SICHUAN MAPO TOFU (2501 - 4000)
CHAR KWAY TEOW (2501 - 4000)
CHRYSANTHEMUM TEA (2501 - 4000)
JAPANESE CHICKEN KATSU CURRY (4001 - 5500)
SPICY TUNA ROLL (4001 - 5500)
GOLDEN BITES CORNER (4001 - 5500)
LYCHEE ROSE TEA (4001 - 5500)
CHAR SIU BAO (4001 - 5500)
CHRYSANTHEMUM TEA (5501 - 7000)
GOLDEN CURRY NOODLES (5501 - 7000)
KOREAN JAPCHAE (5501 - 7000)
KUNG PAO CHICKEN (5501 - 7000)
LYCHEE ROSE TEA (5501 - 7000)
THAI PAD THAI (7001 - 9000)
CHRYSANTHEMUM TEA (7001 - 9000)
LO BAK GO (7001 - 9000)
HAR GOW (7001 - 9000)
KOREAN JAPCHAE (7001 - 9000)
Item Name

This query generates a report detailing the top 5 food and beverage items ordered by different income groups, showing total orders and sales in Malaysian Ringgit (RM). It reveals purchasing patterns across income brackets, where lower-income groups tend to favour affordable items like "Lo Bak Go" and "California Roll," while higher-income groups prefer more unique options. Popular items like "Chrysanthemum Tea" and "Lychee Rose Tea" appear across multiple income groups, suggesting widespread appeal. The report also highlights that some items generate high sales despite fewer orders, indicating higher pricing. Based on this report, businesses can create tailored marketing campaigns, such as promoting cost-effective deals for low-income groups and premium versions of popular items for higher-income groups. Businesses can enhance their visibility by placing these items at the top of the menu in special sections like "Bestsellers". Offering bundled deals can encourage higher-value purchases. Additionally, creating value deals or introducing premium and seasonal variations of these favourites can further boost interest and sales, ensuring the menu caters effectively to diverse customer preferences while leading to revenue growth.

3.5 Vithiya Saraswathi a/p Sockalingam

3.5.1 Top 5 and Least 3 Menu Items Report in a Year with 5% projected sales

SQL:

```
SPOOL 'C:\Users\Windows10\Downloads\DW\output_1.txt'

ACCEPT year_prompt NUMBER PROMPT 'Enter the year (default is 2024): '
DEFINE year = &year_prompt

VARIABLE month1 NUMBER
VARIABLE month2 NUMBER
VARIABLE month3 NUMBER

ACCEPT month1 NUMBER PROMPT 'Enter the first month (e.g., 5 for May): '
ACCEPT month2 NUMBER PROMPT 'Enter the second month (e.g., 6 for June): '
ACCEPT month3 NUMBER PROMPT 'Enter the third month (e.g., 7 for July): '


SET PAGESIZE 50
SET LINESIZE 200
SET COLSEP ' | '

COLUMN ITEM_CATEGORY FORMAT A10 HEADING 'Item|Category'
COLUMN MENUID FORMAT 9999 HEADING 'MenuID'
COLUMN ITEMNAME FORMAT A25 HEADING 'Item|Name'
COLUMN QUANTITY FORMAT 999,999 HEADING 'Quantity'
COLUMN INITIAL_SALES FORMAT 999,999.99 HEADING 'Initial|Sales'
COLUMN INITIAL_CONTRIBUTION_PERCENT FORMAT 999.99 HEADING 'Initial|Contribution|Percent(%)'
COLUMN PROJECTED_SALES FORMAT 999,999.99 HEADING 'Projected|Sales'
COLUMN PROJECTED_CONTRIBUTION_PERCENT FORMAT 999.99 HEADING 'Projected|Contribution|Percent(%)'
COLUMN CONTRIBUTION_PERCENT_VS_TOP FORMAT 999.99 HEADING 'Contribution|Percent|vs|Top(%)'
```

```
TTITLE LEFT SKIP 2 '<< Top 5 and Least 3 Menu Items Report in Year '&year' with 5% projected sales >>'
SKIP 2 LEFT '**Data for Months: '&month1', '&month2', '&month3  ** SKIP 2 LEFT 'PAGE: ' FORMAT 99
SQL.PNO SKIP 2

BREAK ON ITEM_CATEGORY SKIP 1

WITH orders_summary AS (
    SELECT
        m.ITEMNAME,
        m.MENUID,
        SUM(i.QUANTITY) AS total_quantity,
        SUM(i.lineTotal) AS total_sales
    FROM Items_Fact i
    JOIN date_dim d ON i.date_key = d.date_key
    JOIN menu_dim m ON i.menu_key = m.menu_key
    WHERE d.cal_year = &year
      AND d.cal_month_year IN (&month1, &month2, &month3)
    GROUP BY m.ITEMNAME, m.MENUID
),
total_sales_all AS (
    SELECT SUM(total_sales) AS total_revenue FROM orders_summary
),
ranked_items AS (
    SELECT
        ITEMNAME,
        MENUID,
        total_quantity,
        total_sales,
        ROUND((total_sales / (SELECT total_revenue FROM total_sales_all)) * 100, 2) AS
initial_contrib_percent,
        ROW_NUMBER() OVER (ORDER BY total_quantity DESC) AS rank_desc,
        ROW_NUMBER() OVER (ORDER BY total_quantity ASC) AS rank_asc
    FROM orders_summary
```

```
),
top_5_items AS (
    SELECT
        ITEMNAME,
        MENUID,
        total_quantity AS qty_top5,
        total_sales AS sales_top5,
        initial_contrib_percent AS contrib_top5,
        ROUND(total_sales * 1.05, 2) AS proj_sales_top5, -- Projected sales with 5% increase
        ROUND((total_sales * 1.05 / (SELECT total_revenue FROM total_sales_all)) * 100, 2) AS
proj_contrib_top5, -- Projected contribution percentage with 5% increase
        'Top 5' AS item_cat,
        NULL AS qty_least3,
        NULL AS sales_least3,
        NULL AS contrib_least3,
        NULL AS proj_sales_least3,
        NULL AS proj_contrib_least3
    FROM ranked_items
    WHERE rank_desc <= 5
),
least_3_items AS (
    SELECT
        ITEMNAME,
        MENUID,
        NULL AS qty_top5,
        NULL AS sales_top5,
        NULL AS contrib_top5,
        NULL AS proj_sales_top5,
        NULL AS proj_contrib_top5,
        'Least 3' AS item_cat,
        total_quantity AS qty_least3,
        total_sales AS sales_least3,
        initial_contrib_percent AS contrib_least3,
        NULL AS proj_sales_least3,
```

```
        NULL AS proj_contrib_least3
    FROM ranked_items
    WHERE rank_asc <= 3
),
first_top_5_sales AS (
    SELECT sales_top5 AS first_top5_sales
    FROM top_5_items
    WHERE ROWNUM = 1
),
combined_data AS (
    SELECT * FROM top_5_items
    UNION ALL
    SELECT * FROM least_3_items
)
SELECT
    item_cat AS item_category,
    MENUID,
    ITEMNAME,
    COALESCE(qty_top5, qty_least3) AS quantity,
    COALESCE(sales_top5, sales_least3) AS initial_sales,
    COALESCE(contrib_top5, contrib_least3) AS initial_contribution_percent,
    COALESCE(proj_sales_top5, proj_sales_least3) AS projected_sales,
    COALESCE(proj_contrib_top5, proj_contrib_least3) AS projected_contribution_percent,
    CASE
        WHEN item_cat = 'Least 3' THEN
            ROUND((sales_least3 / (SELECT first_top5_sales FROM first_top_5_sales)) * 100, 2)
        ELSE
            NULL
    END AS contribution_percent_vs_top
FROM combined_data
ORDER BY
    CASE
        WHEN item_cat = 'Top 5' THEN 1
        WHEN item_cat = 'Least 3' THEN 2
```

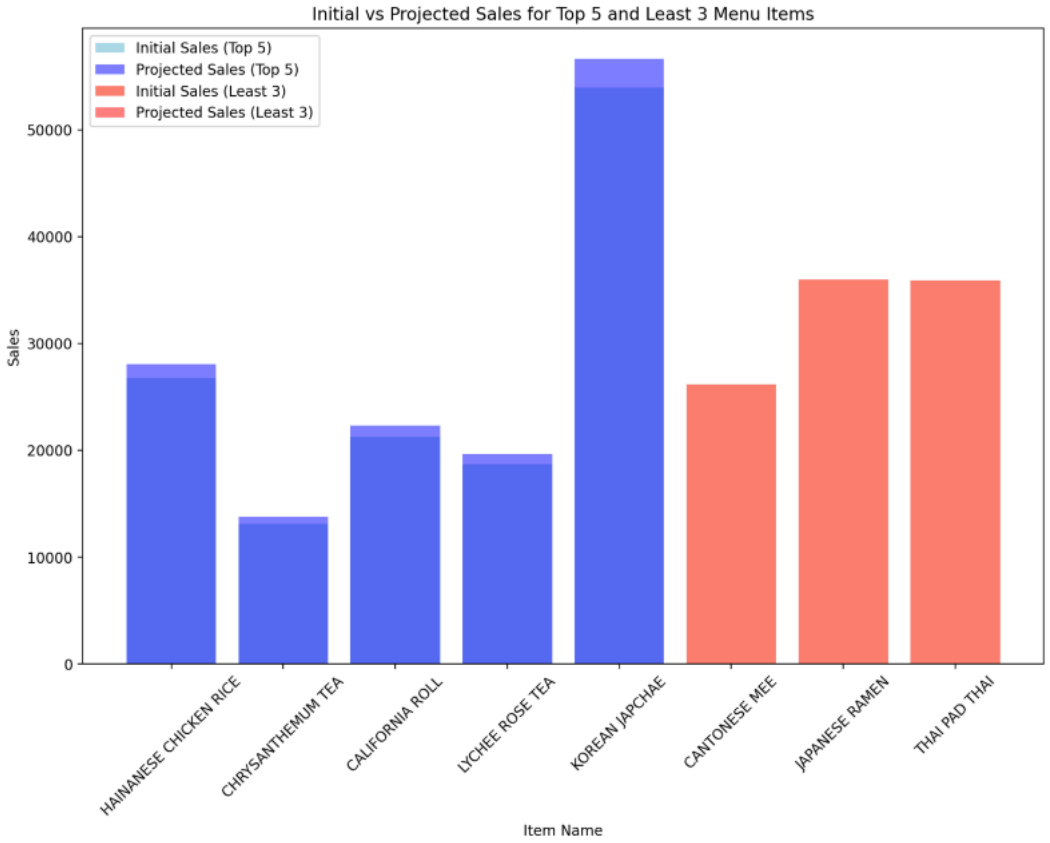
```
    END,
    COALESCE(qty_top5, qty_least3) DESC;


CLEAR COLUMNS
CLEAR BREAKS
TTITLE OFF;

SPOOL OFF
```

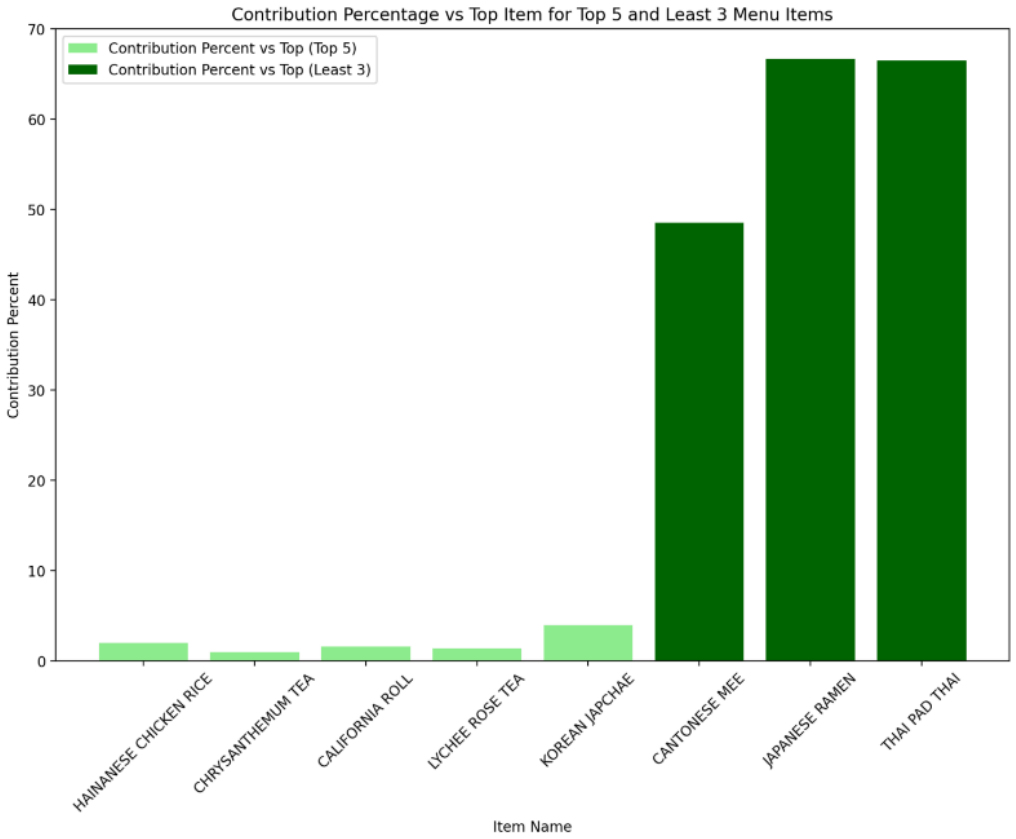
OUTPUT:

```
<< Top 5 and Least 3 Menu Items Report in Year 2024 with 5% projected sales >>

**Data for Months: 5, 6, 7**

PAGE:   1
```

Item Category	MenuID	Item Name	Quantity	Initial Sales	Initial Contribution Percent(%)	Projected Sales	Projected Contribution Percent(%)	Contribution Percent vs Top(%)
Top 5	1021	HAINANESE CHICKEN RICE	4,456	26,736.00	1.89	28,072.80	1.99	
	1040	CHRYSANTHEMUM TEA	4,371	13,113.00	.93	13,768.65	.97	
	1047	CALIFORNIA ROLL	4,247	21,235.00	1.50	22,296.75	1.58	
	1028	LYCHEE ROSE TEA	4,160	18,720.00	1.32	19,656.00	1.39	
	1046	KOREAN JAPCHAE	4,150	53,950.00	3.82	56,647.50	4.01	
Least 3	1041	CANTONESE MEE	3,273	26,184.00	1.85			48.53
	1016	JAPANESE RAMEN	3,270	35,970.00	2.55			66.67
	1026	THAI PAD THAI	3,262	35,882.00	2.54			66.51

```
8 rows selected.
```


Initial vs Projected Sales for Top 5 and Least 3 Menu Items
Initial Sales (Top 5)
Projected Sales (Top 5)
Initial Sales (Least 3)
Projected Sales (Least 3)
Sales
0
10000
20000
30000
40000
50000
HAINANESE CHICKEN RICE
CHRYSANTHEMUM TEA
CALIFORNIA ROLL
LYCHEE ROSE TEA
KOREAN JAPCHAE
CANTONESE MEE
JAPANESE RAMEN
THAI PAD THAI
Item Name



This query analyses menu item performance for 2024 across three specified months, focusing on the top 5 best-selling and least 3 selling items. It calculates total quantity sold, sales, and contribution percentages for each item. For the top 5, the query projects sales and contribution assuming a 5% price increase, while the least 3 are compared against the top item to assess potential removal. Items with projected contribution percentages above the 0.05 threshold are candidates for a price increase, while items with contributions below 50% of the top-selling item may be removed. Based on the analysis, Hainanese Chicken Rice, California Roll, Lychee Rose Tea, and Korean Japchae qualify for a price increase. Chrysanthemum Tea falls below the threshold, so its price should remain unchanged. Cantonese Mee has a low contribution and could be removed, but Japanese Ramen and Thai Pad Thai perform well enough to stay.

3.5.2 Comparison Sales of Weekdays and Weekends based on May, June and July in 2019

SQL:

```
SPOOL 'C:\Users\Windows10\Downloads\DW\output_2.txt'

ACCEPT year_prompt NUMBER PROMPT 'Enter the year: '
DEFINE year = &year_prompt

VARIABLE month1 NUMBER
VARIABLE month2 NUMBER
VARIABLE month3 NUMBER

ACCEPT month1 NUMBER PROMPT 'Enter the first month (e.g., 5 for May): '
ACCEPT month2 NUMBER PROMPT 'Enter the second month (e.g., 6 for June): '
ACCEPT month3 NUMBER PROMPT 'Enter the third month (e.g., 7 for July): '

SET PAGESIZE 50
SET LINESIZE 110
SET COLSEP ' | '

COLUMN cal_year FORMAT 9999 HEADING 'Year'
COLUMN cal_month_year FORMAT 99 HEADING 'Month'
COLUMN day_type FORMAT A8 HEADING 'Day Type'
COLUMN num_of_days FORMAT 999 HEADING 'No. of Days'
COLUMN TOTAL_REVENUE_wholeday FORMAT 999,999,999.99 HEADING 'Total Revenue'
COLUMN TOTAL_REVENUE_loss FORMAT 999,999,999.99 HEADING 'Revenue Loss'
COLUMN Avg_loss_hr_dy FORMAT 999,999.99 HEADING 'Avg Loss/hr'
COLUMN Percentage_of_Loss FORMAT 999.99 HEADING 'Loss (%)'
```

```
TTITLE LEFT SKIP 2 '<< Comparison Sales of Weekdays and Weekends in '&year': Decision to Remain Open
or Close an Hour Early>>' SKIP 2 LEFT '**Data for Months: '&month1', '&month2', '&month3  ** SKIP 2
LEFT 'PAGE: ' FORMAT 99 SQL.PNO SKIP 2

BREAK ON cal_year SKIP 2
BREAK ON cal_month_year SKIP 1

-- Total Sales (including both weekday and weekend sales)
CREATE OR REPLACE VIEW Total_Sales AS
SELECT
    D.cal_year,
    D.cal_month_year,
    'All Days' AS day_type,
    COUNT(DISTINCT D.date_key) AS num_of_days,
    SUM(CASE WHEN A.orderTime BETWEEN '09:36' AND '22:00' THEN A.orderAmount ELSE 0 END) AS
TOTAL_REVENUE_wholeday,
    SUM(CASE WHEN A.orderTime BETWEEN '21:00' AND '22:00' THEN A.orderAmount ELSE 0 END) AS
TOTAL_REVENUE_loss,
    CASE
        WHEN COUNT(DISTINCT D.date_key) = 0 THEN 0
        ELSE SUM(CASE WHEN A.orderTime BETWEEN '21:00' AND '22:00' THEN A.orderAmount ELSE 0 END) /
COUNT(DISTINCT D.date_key)
    END AS Avg_loss_hr_dy,
    CASE
        WHEN SUM(CASE WHEN A.orderTime BETWEEN '09:36' AND '22:00' THEN A.orderAmount ELSE 0 END) = 0
THEN 0
        ELSE
            (SUM(CASE WHEN A.orderTime BETWEEN '21:00' AND '22:00' THEN A.orderAmount ELSE 0 END) /
            SUM(CASE WHEN A.orderTime BETWEEN '09:36' AND '22:00' THEN A.orderAmount ELSE 0 END)) *
100
    END AS Percentage_of_Loss
FROM Orders_Fact A
JOIN date_dim D ON A.date_key = D.date_key
WHERE D.cal_year = &year AND D.cal_month_year IN (&month1, &month2, &month3)
```

```
GROUP BY D.cal_year, D.cal_month_year
ORDER BY D.cal_year, D.cal_month_year;

-- Weekday Sales
CREATE OR REPLACE VIEW Week_Day_Sales AS
SELECT
    D.cal_year,
    D.cal_month_year,
    'Weekday' AS day_type,
    COUNT(DISTINCT D.date_key) AS num_of_days,
    SUM(CASE WHEN A.orderTime BETWEEN '09:36' AND '22:00' THEN A.orderAmount ELSE 0 END) AS
TOTAL_REVENUE_wholeday,
    SUM(CASE WHEN A.orderTime BETWEEN '21:00' AND '22:00' THEN A.orderAmount ELSE 0 END) AS
TOTAL_REVENUE_loss,
    CASE
        WHEN COUNT(DISTINCT D.date_key) = 0 THEN 0
        ELSE SUM(CASE WHEN A.orderTime BETWEEN '21:00' AND '22:00' THEN A.orderAmount ELSE 0 END) /
COUNT(DISTINCT D.date_key)
    END AS Avg_loss_hr_dy,
    CASE
        WHEN SUM(CASE WHEN A.orderTime BETWEEN '09:36' AND '22:00' THEN A.orderAmount ELSE 0 END) = 0
THEN 0
        ELSE
            (SUM(CASE WHEN A.orderTime BETWEEN '21:00' AND '22:00' THEN A.orderAmount ELSE 0 END) /
            SUM(CASE WHEN A.orderTime BETWEEN '09:36' AND '22:00' THEN A.orderAmount ELSE 0 END)) *
100
    END AS Percentage_of_Loss
FROM Orders_Fact A
JOIN date_dim D ON A.date_key = D.date_key
WHERE D.cal_year = &year AND D.cal_month_year IN (&month1, &month2, &month3) AND D.WEEKDAY_IND = 'Y'
GROUP BY D.cal_year, D.cal_month_year
ORDER BY D.cal_year, D.cal_month_year;

-- Weekend Sales
```

```
CREATE OR REPLACE VIEW Week_End_Sales AS
SELECT
    D.cal_year,
    D.cal_month_year,
    'Weekend' AS day_type,
    COUNT(DISTINCT D.date_key) AS num_of_days,
    SUM(CASE WHEN A.orderTime BETWEEN '09:36' AND '22:00' THEN A.orderAmount ELSE 0 END) AS
TOTAL_REVENUE_wholeday,
    SUM(CASE WHEN A.orderTime BETWEEN '21:00' AND '22:00' THEN A.orderAmount ELSE 0 END) AS
TOTAL_REVENUE_loss,
    CASE
        WHEN COUNT(DISTINCT D.date_key) = 0 THEN 0
        ELSE SUM(CASE WHEN A.orderTime BETWEEN '21:00' AND '22:00' THEN A.orderAmount ELSE 0 END) /
COUNT(DISTINCT D.date_key)
    END AS Avg_loss_hr_dy,
    CASE
        WHEN SUM(CASE WHEN A.orderTime BETWEEN '09:36' AND '22:00' THEN A.orderAmount ELSE 0 END) = 0
THEN 0
        ELSE
            (SUM(CASE WHEN A.orderTime BETWEEN '21:00' AND '22:00' THEN A.orderAmount ELSE 0 END) /
            SUM(CASE WHEN A.orderTime BETWEEN '09:36' AND '22:00' THEN A.orderAmount ELSE 0 END)) *
100
    END AS Percentage_of_Loss
FROM Orders_Fact A
JOIN date_dim D ON A.date_key = D.date_key
WHERE D.cal_year = &year AND D.cal_month_year IN (&month1, &month2, &month3) AND D.WEEKDAY_IND = 'N'
GROUP BY D.cal_year, D.cal_month_year
ORDER BY D.cal_year, D.cal_month_year;

-- Final query combining Total Sales, Weekday Sales, and Weekend Sales
SELECT
    cal_year,
    cal_month_year,
    day_type,
```

```
    num_of_days,
    TOTAL_REVENUE_wholeday,
    TOTAL_REVENUE_loss,
    Avg_loss_hr_dy,
    Percentage_of_Loss
FROM (
    SELECT * FROM Week_Day_Sales
    UNION ALL
    SELECT * FROM Week_End_Sales
)
ORDER BY cal_year, cal_month_year, day_type;

CLEAR COLUMNS
CLEAR BREAKS
TTITLE OFF;

SPOOL OFF
```

OUTPUT:

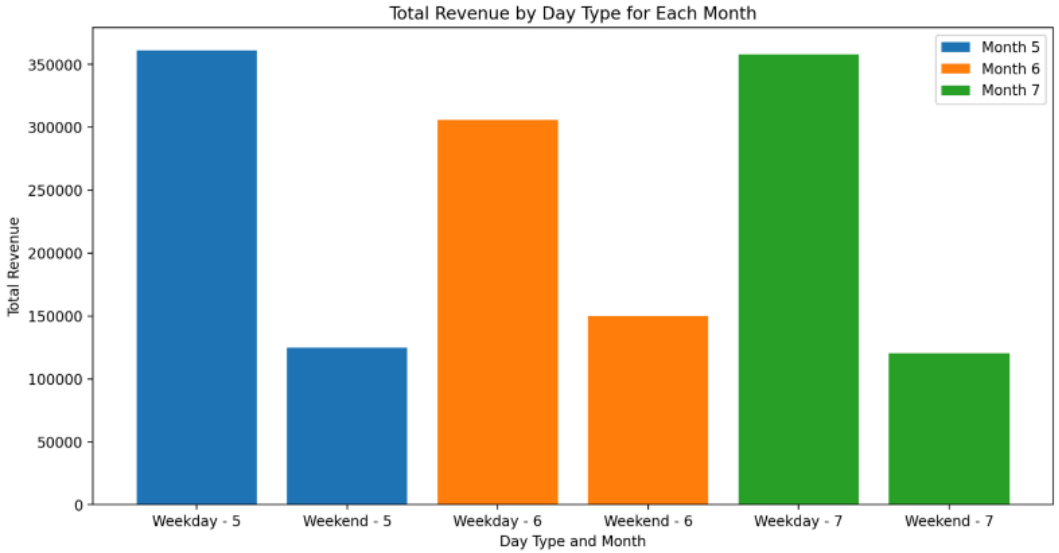
<< Comparison Sales of Weekdays and Weekends in 2019: Decision to Remain Open or Close an Hour Early>>

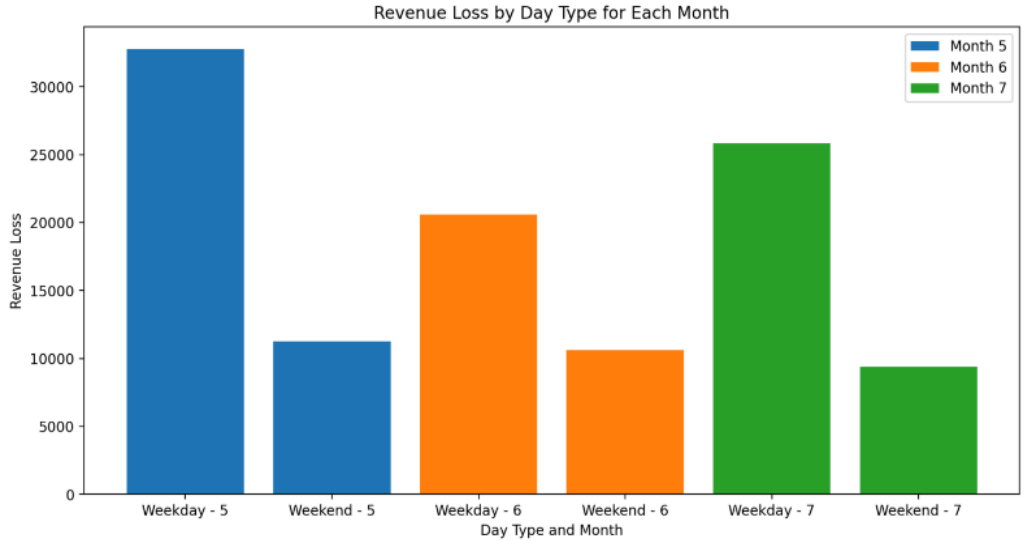
Data for Months: 5, 6, 7

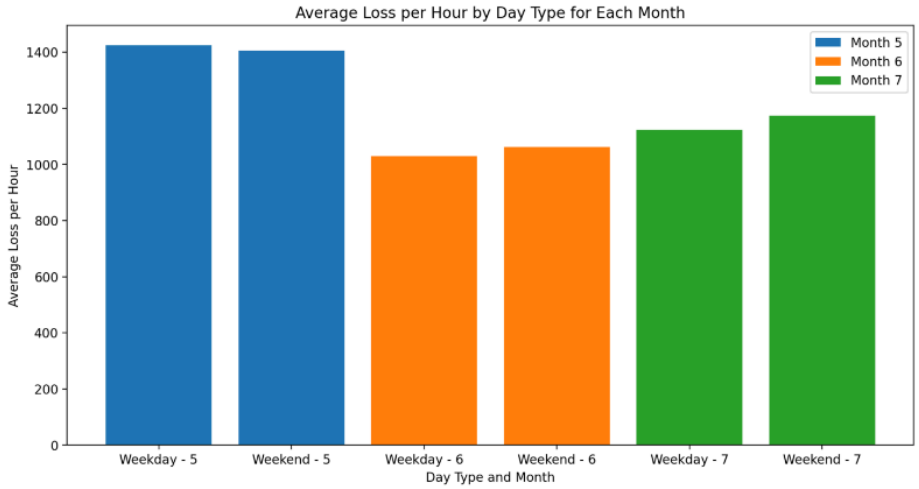
PAGE: 1

Year	Month	Day Type	No. of Days	Total Revenue	Revenue Loss	Avg Loss/hr	Loss (%)
2019	5	Weekday	23	361,024.00	32,750.50	1,423.93	9.07
2019		Weekend	8	124,901.00	11,245.50	1,405.69	9.00
2019	6	Weekday	20	305,761.50	20,581.00	1,029.05	6.73
2019		Weekend	10	149,965.00	10,627.50	1,062.75	7.09
2019	7	Weekday	23	357,816.00	25,856.00	1,124.17	7.23
2019		Weekend	8	120,237.50	9,392.50	1,174.06	7.81

6 rows selected.


Total Revenue by Day Type for Each Month
Month 5
Month 6
Month 7
Total Revenue
0
50000
100000
150000
200000
250000
300000
350000
Weekday - 5
Weekend - 5
Weekday - 6
Weekend - 6
Weekday - 7
Weekend - 7
Day Type and Month


Revenue Loss by Day Type for Each Month
Month 5
Month 6
Month 7
Revenue Loss
0
5000
10000
15000
20000
25000
30000
Weekday - 5
Weekend - 5
Weekday - 6
Weekend - 6
Weekday - 7
Weekend - 7
Day Type and Month



This query is designed to compare weekday and weekend sales across May, June, and July 2019 to assess the viability of closing one hour early as a cost-saving measure. The analysis calculates total daily revenue and the revenue lost during the last operational hour, offering insights into how much of the day's sales occur during this final hour. By comparing the revenue loss against a 10% threshold, we can determine if it's financially reasonable to close early on weekdays, weekends, or both. The analysis shows that, for May, June, and July 2019, revenue losses on both weekdays and weekends are within the acceptable 10% limit. However, weekdays typically exhibit higher percentages of revenue loss compared to weekends. This suggests that it would be more strategic to consider closing early on weekdays, where the financial impact of lost sales is higher, but still within the defined threshold. As a result, closing early on weekdays emerges as a viable option for reducing operational costs while keeping revenue losses within acceptable limits. Based on this analysis, the conclusion is that closing early on weekdays would be the more effective strategy for controlling costs.

3.5.3 Christmas vs Non-Christmas Sales Report

SQL:

```
SPOOL 'C:\Users\Windows10\Downloads\DW\output_3.txt'

SET PAGESIZE 50
SET LINESIZE 150
SET COLSEP ' | '

COLUMN cal_year FORMAT 9999 HEADING 'Year'
COLUMN xmas_sales FORMAT 999,999.99 HEADING 'Xmas Sales'
COLUMN avg_non_xmas_sales FORMAT 999,999.99 HEADING 'Avg Non-Xmas|Sales'
COLUMN non_xmas_days FORMAT 999 HEADING 'Non-Xmas|Days'
COLUMN sales_diff FORMAT 999,999.99 HEADING 'Sales Diff'
COLUMN percentage_of_xmas_to_non_xmas FORMAT 999.99 HEADING 'Percentage|of Xmas to|Non-Xmas'

TTITLE LEFT SKIP 2 '<< Christmas vs Non-Christmas Sales Comparison: Decision to Remain Open or Close
>>' SKIP 2 LEFT '**Data for Years: 2023 & 2024**' SKIP 2 LEFT 'PAGE: ' FORMAT 99 SQL.PNO SKIP 2

WITH Sales_Data AS (
    SELECT
        d.cal_year,
        d.festive_season,
        COUNT(DISTINCT d.date_key) AS num_days,
        SUM(o.orderAmount) AS total_sales
    FROM Orders_Fact o
    JOIN date_dim d ON o.date_key = d.date_key
    WHERE d.cal_year IN (EXTRACT(YEAR FROM TRUNC(SYSDATE)), EXTRACT(YEAR FROM TRUNC(SYSDATE)) - 1)
    GROUP BY d.cal_year, d.festive_season
),
Comparison AS (
    SELECT
        cal_year,
```

```
        MAX(CASE WHEN festive_season = 'Christmas' THEN total_sales ELSE NULL END) AS xmas_sales,
        AVG(CASE WHEN festive_season IS NULL THEN total_sales / num_days ELSE NULL END) AS
avg_non_xmas_sales,
        MAX(CASE WHEN festive_season IS NULL THEN num_days ELSE NULL END) AS non_xmas_days
    FROM Sales_Data
    GROUP BY cal_year
)
SELECT
    cal_year,
    xmas_sales,
    avg_non_xmas_sales,
    non_xmas_days,
    (avg_non_xmas_sales - xmas_sales) AS sales_diff,
    CASE
        WHEN avg_non_xmas_sales = 0 THEN 0
        ELSE ROUND(
            ((xmas_sales / avg_non_xmas_sales) * 100), 2
        )
    END AS percentage_of_xmas_to_non_xmas,
    CASE
        WHEN ((xmas_sales / avg_non_xmas_sales) * 100) > 80 THEN 'Keep Open'
        ELSE 'Consider Closing'
    END AS recommendation
FROM Comparison
ORDER BY cal_year;

CLEAR COLUMNS
CLEAR BREAKS
TTITLE OFF;

SPOOL OFF
```

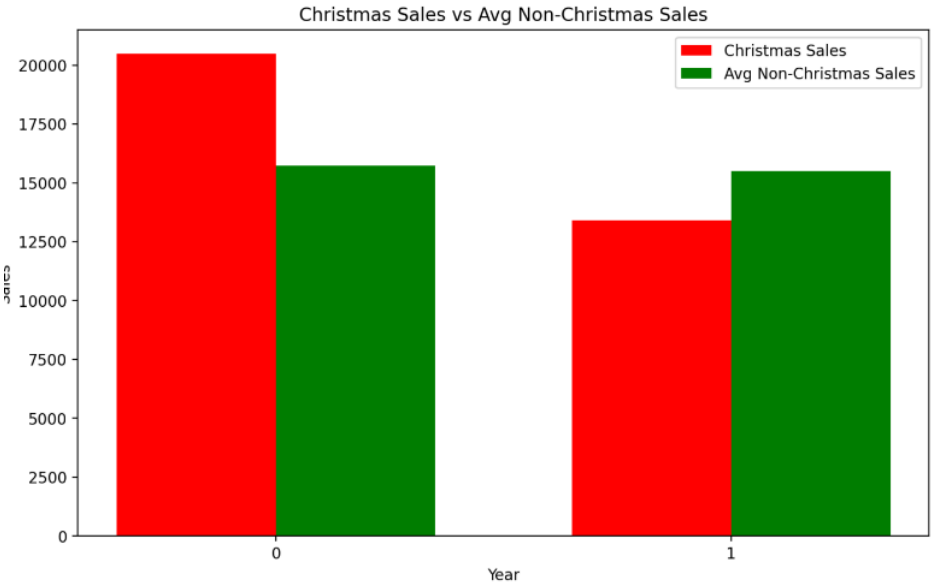
OUTPUT:

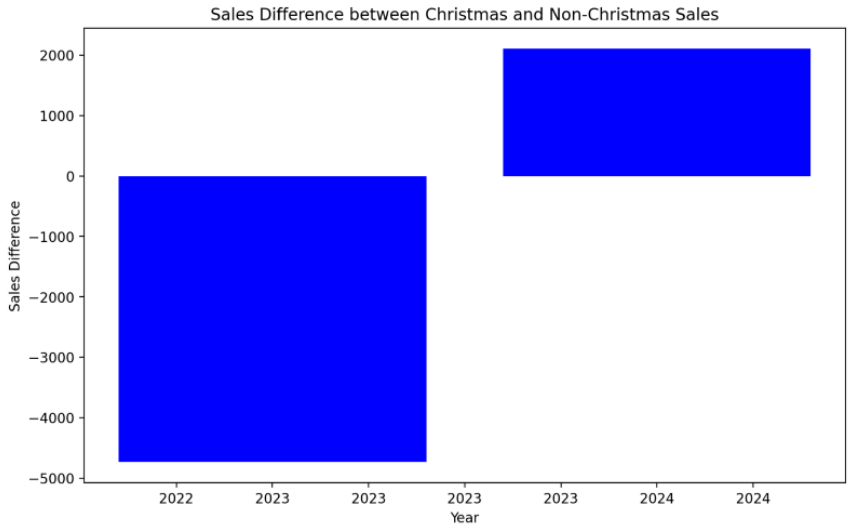
<< Christmas vs Non-Christmas Sales Comparison: Decision to Remain Open or Close >>

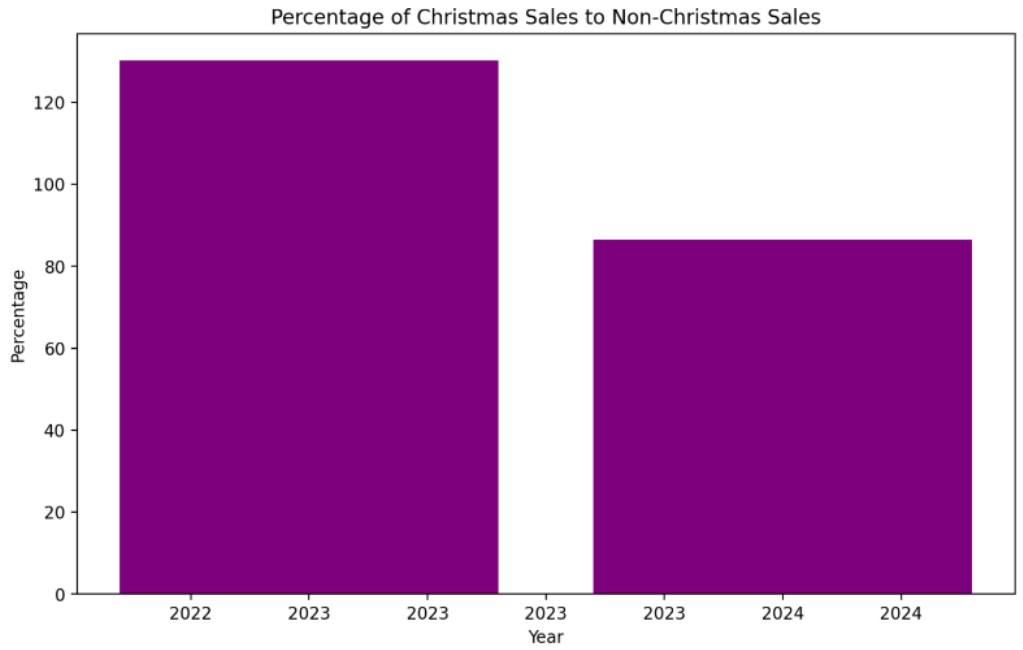
Data for Years: 2023 & 2024

PAGE: 1

Year	Xmas Sales	Avg Non-Xmas Sales	Non-Xmas Days	Sales Diff	Percentage of Xmas to Non-Xmas	RECOMMENDATION
2023	20,467.50	15,731.52	361	-4,735.98	130.11	Keep Open
2024	13,401.50	15,502.16	362	2,100.66	86.45	Keep Open


Christmas Sales vs Avg Non-Christmas Sales
Christmas Sales
Avg Non-Christmas Sales
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
Year


Sales Difference between Christmas and Non-Christmas Sales
Sales Difference
2000
1000
0
−1000
−2000
−3000
−4000
−5000
2022
2023
2023
2023
2023
2024
2024
Year



This query compares Christmas sales with average non-Christmas sales for 2023 and 2024 to determine whether it's profitable to stay open during the Christmas season. The analysis highlights key insights into whether the store should remain open during the holidays. In 2023, the store generated $20,467.50 in Christmas sales, which amounted to 130.11% of the average daily non-Christmas sales ($15,731.52). This figure greatly exceeds the 80% threshold, indicating that Christmas sales were significantly higher than the average daily non-Christmas sales. The fact that Christmas sales were 30.11% higher than typical non-Christmas days led to a clear recommendation to "Keep Open" during Christmas due to the substantial sales boost. In 2024, Christmas sales were slightly lower at $13,401.50, representing 86.45% of the average daily non-Christmas sales ($15,502.16). While Christmas sales in 2024 were below the non-Christmas average, they still exceeded the 80% threshold, suggesting that staying open was still profitable enough. As a result, the recommendation for 2024 is also to "Keep Open". Given that in both years, Christmas sales exceeded the 80% threshold, the decision is made to remain open for the next Christmas season as well, based on this forward planning.