BAIT3003 Data Warehouse

Assignment
2024

Programme : RDS2Y2S3

Tutorial Group : 3

Date Submitted to Tutor : 15 September 2024

Team Members :

No	Student Name	Student ID
1.	Yam Jason	22WMR13662
2.	Wong Yee En	22WMR13659
3.	Ashantha Rosary James K Arokiasamy	22WMR14161
4.	Wong Zi Ning	22WMR13661
5.	Tan Wan Yin	22WMR14787

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 (9-14) · Poor (0-8)		1
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

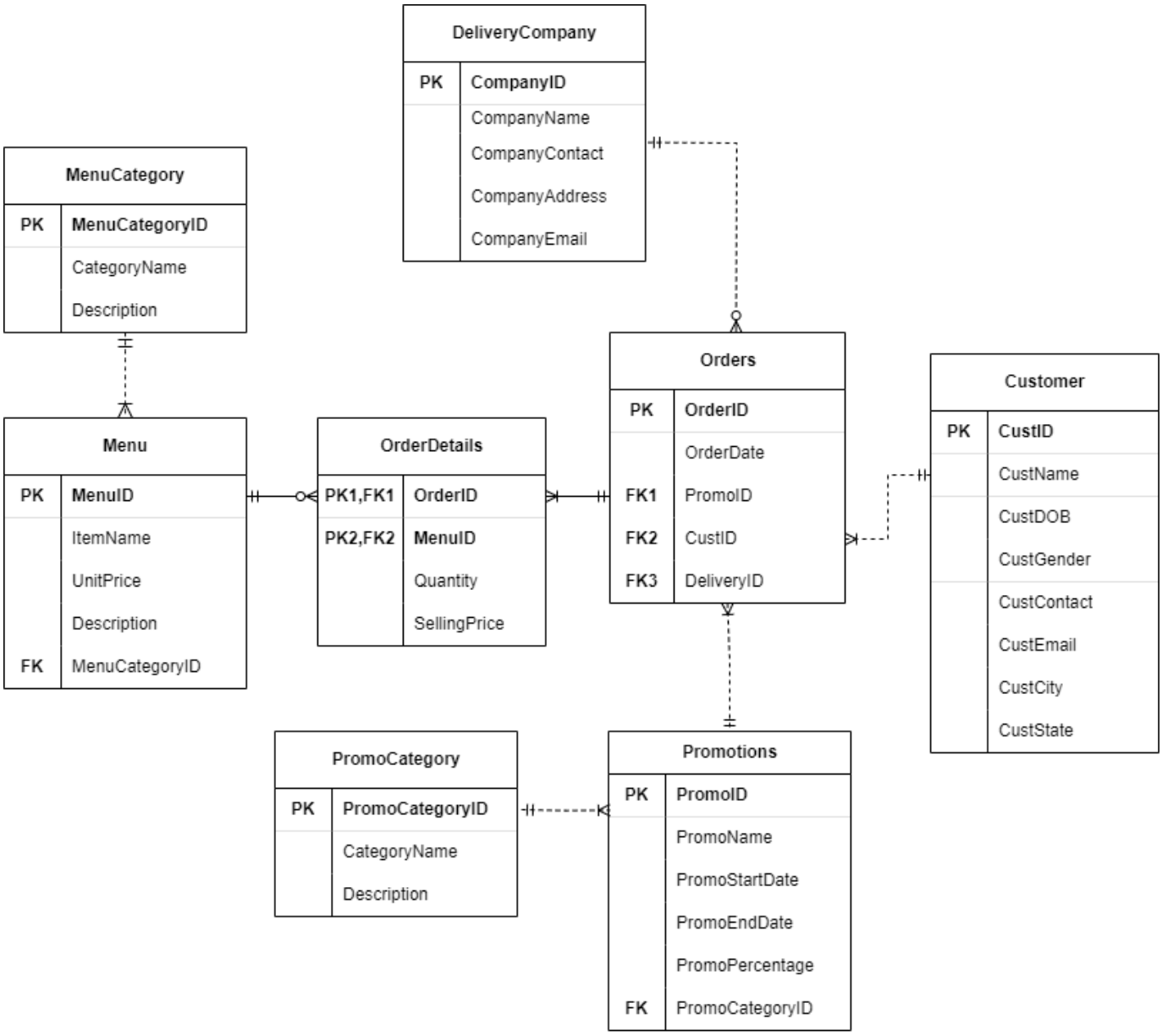
Student Name	Task 3	Total Marks
Yam Jason		
Wong Yee En		
Ashantha Rosary James K Arokiasamy		
Wong Zi Ning		
Tan Wan Yin		

Table Of Contents

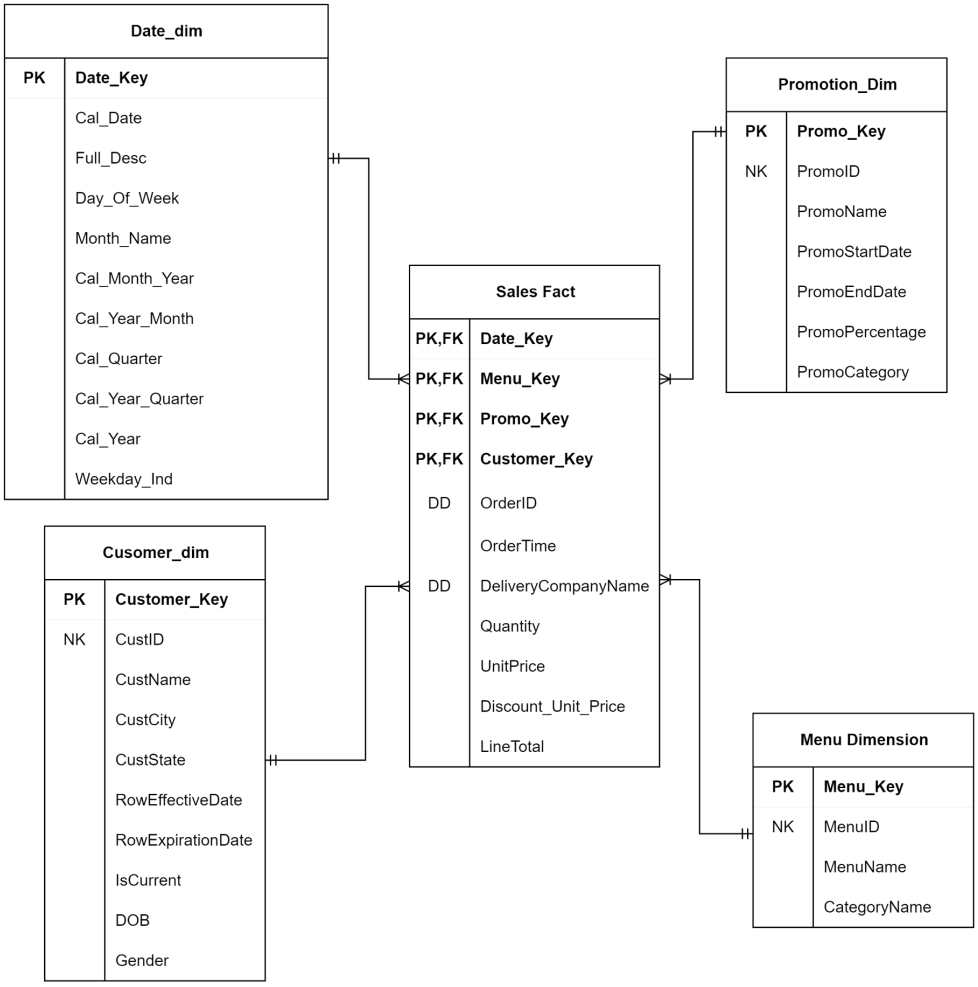
Chapter 1 : Design of Data Warehouse

1.1 Logical Design

1.1.1 Original Database



1.1.2 Star Schema Dimension and Fact Tables



1.2 Physical Design

1.2.1 Dimension Tables

Menu Dimension Table

```
Create table menu_dim
(menu_key      NUMBER        NOT NULL,
 menuID        NUMBER        NOT NULL,
 menuName      VARCHAR(40)   NOT NULL,
 categoryName  VARCHAR(15)   NOT NULL,
 Constraint PK_menu_key primary key(menu_key)
);
```

Date Dimension Table

```
Create table Date_dim
(date_key            number not null, -- running number
 cal_date            date not null,   -- all the dates in the
calendar
 full_desc           varchar(40),     -- spelling description of date
 day_of_week         number(1),       -- 1 to 7
 month_name          varchar(9),      -- 'January' to 'December'
 cal_month_year      number(2),       -- 1 to 12
 cal_year_month      char(7),      -- e.g. '2024-07'
 cal_quarter         char(2),         -- 'Q1' to 'Q4'
 cal_year_quarter    char(7),         -- e.g. '2024-Q1'
 cal_year            number(4),
 weekday_ind         char(1),         -- 'Y'/'N'
constraint PK_date_key primary key(date_key)
);
```

Promotion Dimension Table

```
Create table promotion_dim
(promo_key       NUMBER        NOT NULL,
 promoID         NUMBER        NOT NULL,
 promoName       VARCHAR(40)   NOT NULL,
 promoStartDate  DATE          NOT NULL,
 promoEndDate    DATE          NOT NULL,
 promoPercentage NUMBER        NOT NULL,
 promoCategory   VARCHAR(15)   NOT NULL,
 Constraint PK_promo_key primary key(promo_key)
);
```

Customer Dimension Table

```
Create table customer_dim
(customer_key       number not null,
  CustID            number NOT NULL,
  CustName          VARCHAR(30) NOT NULL,
  CustCity          VARCHAR(20),
  CustState         VARCHAR(20),
  RowEffectiveDate  Date    DEFAULT DATE '2014-06-01' NOT NULL,
  RowExpirationDate Date    DEFAULT DATE '9999-12-31' NOT NULL,
  isCurrent         Char(1) DEFAULT 'Y' NOT NULL, --Y/N
constraint PK_Cust_key primary key (customer_key)
);
```

1.2.2 Fact Table

```
CREATE TABLE SalesFact
( DATE_KEY                NUMBER      NOT NULL,
  MENU_KEY                NUMBER      NOT NULL,
  PROMO_KEY               NUMBER      NOT NULL,
  CUSTOMER_KEY            NUMBER      NOT NULL,
  ORDERID                 NUMBER      NOT NULL,
  OrderTime               CHAR(5)     NOT NULL,
  DeliveryCompanyName     VARCHAR(18) NOT NULL,
  Quantity                NUMBER      NOT NULL,
  UnitPrice              NUMBER(4,2) NOT NULL,
  Discount_Unit_Price     NUMBER(4,2) NOT NULL,
  LineTotal               NUMBER(6,2) NOT NULL,
  CONSTRAINT PK_FACT PRIMARY KEY (DATE_KEY, MENU_KEY, PROMO_KEY,
CUSTOMER_KEY, ORDERID),
  CONSTRAINT FK_DATE_KEY FOREIGN KEY (DATE_KEY) REFERENCES DATE_DIM,
  CONSTRAINT FK_MENU_KEY FOREIGN KEY (MENU_KEY) REFERENCES MENU_DIM,
  CONSTRAINT FK_PROMO_KEY FOREIGN KEY (PROMO_KEY) REFERENCES
PROMOTION_DIM,
  CONSTRAINT FK_CUST_KEY FOREIGN KEY (CUSTOMER_KEY) REFERENCES
CUSTOMER_DIM
);
```

Chapter 2 : Extract, Transform, Load, Process

2.1 Script for initial loading

Menu Dimension Table

```
drop sequence menu_dim_seq;
create sequence menu_dim_seq
  start with 1001;

insert into menu_dim
Select menu_dim_seq.nextval,
       M.menuID,
       UPPER(M.ItemName),
       UPPER(C.CategoryName)
FROM menu M
Join menuCategory C on M.menuCategoryID = C.menuCategoryID;
```

Date Dimension Table

```
drop sequence date_seq;
create sequence date_seq
  start with 100001
  increment by 1;

CREATE OR REPLACE VIEW v_invalid_dates AS
SELECT *
FROM Date_dim
WHERE cal_date IS NULL
   OR day_of_week NOT BETWEEN 1 AND 7
   OR cal_month_year NOT BETWEEN 1 AND 12
   OR weekday_ind IS NULL;


SET SERVEROUTPUT ON
DROP PROCEDURE prc_initial_load_date_dim;

CREATE OR REPLACE PROCEDURE prc_initial_load_date_dim AS
BEGIN
  DECLARE
    startDate DATE := TO_DATE('01/07/2014','DD/MM/YYYY');
    endDate DATE := TO_DATE('30/06/2024','DD/MM/YYYY');
    v_CAL_DATE DATE;
    v_FULL_DESC VARCHAR2(40);
    v_DAY_OF_WEEK NUMBER(1);
    v_MONTH_NAME VARCHAR2(9);
    v_CAL_MONTH_YEAR NUMBER(2);
    v_CAL_YEAR_MONTH CHAR(7);
```

```
    v_CAL_QUARTER CHAR(2);
    v_CAL_YEAR_QUARTER CHAR(7);
    v_CAL_YEAR NUMBER(4);
    v_WEEKDAY_IND CHAR(1);

  BEGIN
    WHILE (startDate <= endDate) LOOP
      v_CAL_DATE := startDate;
      v_FULL_DESC := TO_CHAR(startDate, 'DD') || ' Of ' ||
                     TO_CHAR(startDate, 'Month') || ' ' ||
                     TO_CHAR(startDate, 'Year');
      v_DAY_OF_WEEK := TO_CHAR(startDate, 'D');
      v_MONTH_NAME        := to_char(startDate,'MONTH');
      v_CAL_MONTH_YEAR    := to_char(startDate,'MM');
      v_CAL_QUARTER := 'Q' || TO_CHAR(startDate, 'Q');
      v_CAL_YEAR := TO_CHAR(startDate, 'YYYY');
      v_CAL_YEAR_MONTH := v_CAL_YEAR || '-' || v_CAL_MONTH_YEAR;
      v_CAL_YEAR_QUARTER := v_CAL_YEAR || '-' || v_CAL_QUARTER;

      IF (v_DAY_OF_WEEK BETWEEN 2 AND 6) THEN
        v_WEEKDAY_IND := 'Y';
      ELSE
        v_WEEKDAY_IND := 'N';
           END IF;
      -- Additional data validation (scrubbing)
           IF v_CAL_DATE IS NULL OR v_CAL_YEAR < 2014 THEN
        DBMS_OUTPUT.PUT_LINE('Invalid date detected: ' ||
v_CAL_DATE);
        CONTINUE;
      END IF;

      -- Insert data into Date_dim table
      BEGIN
        INSERT INTO Date_dim VALUES (
          date_seq.NEXTVAL,
          v_CAL_DATE,
          v_FULL_DESC,
          v_DAY_OF_WEEK,
          v_MONTH_NAME,
           v_CAL_MONTH_YEAR,
          v_CAL_YEAR_MONTH,
          v_CAL_QUARTER,
          v_CAL_YEAR_QUARTER,
          v_CAL_YEAR,
          v_WEEKDAY_IND
```

```
        );
      EXCEPTION
        WHEN OTHERS THEN
          DBMS_OUTPUT.PUT_LINE('Error occurred while inserting data:
|| SQLERRM);
          ROLLBACK;
      END;

      -- Increment the start date
      startDate := startDate + 1;
    END LOOP;
  END;

  DBMS_OUTPUT.PUT_LINE('ETL process completed successfully.');
END;
/

EXEC prc_initial_load_date_dim;
select count(*) from date_dim;
SELECT * FROM v_invalid_dates;
```

Promotion Dimension Table

```
drop sequence promotion_dim_seq;
create sequence promotion_dim_seq
  start with 1001;

insert into promotion_dim
Select promotion_dim_seq.nextval,
       p.promoID,
       UPPER(p.promoName),
       P.promoStartDate,
       P.promoEndDate,
       P.promoPercentage,
       UPPER(C.CategoryName)
FROM promotions P
Join promoCategory C on P.promoCategoryID = C.promoCategoryID;
```

Customer Dimension Table

```
drop sequence cust_dim_seq;
create sequence cust_dim_seq
   start with 1001;

INSERT INTO customer_dim (customer_key, CustID, CustName, CustCity,
CustState)
```

```
SELECT cust_dim_seq.NEXTVAL,
       CUSTID,
       UPPER(CustName),
       UPPER(CustCity),
       UPPER(CustState)
FROM new_cust;

DROP SEQUENCE dob_seq;
CREATE SEQUENCE dob_seq
  START WITH 1001;

DROP TABLE DOB_data;
CREATE TABLE DOB_data (
    record_id NUMBER NOT NULL,
    birth_date DATE,
    PRIMARY KEY(record_id)
);

DECLARE
    startdate DATE := TO_DATE('01/01/1954', 'dd/mm/yyyy');
    enddate DATE := TO_DATE('01/01/2003', 'dd/mm/yyyy');
BEGIN
    WHILE startdate <= enddate LOOP
        INSERT INTO DOB_data (record_id, birth_date)
        VALUES (dob_seq.NEXTVAL, startdate);
        startdate := startdate + 1;
    END LOOP;
END;
/

-- Ensure the customer_dim table is ready for updates
ALTER TABLE customer_dim
    ADD (dob DATE DEFAULT TO_DATE('01/01/1999', 'dd/mm/yyyy'),
         gender CHAR(1) DEFAULT 'M');

-- Initial update of DOB and Gender
DECLARE
    CURSOR cust_cur IS
        SELECT customer_key FROM customer_dim;

    offset1 NUMBER;
    offset2 NUMBER;
    v_gender CHAR(1);

BEGIN
```

```
    FOR cust_rec IN cust_cur LOOP
        v_gender := 'M';
        offset1 := TRUNC(DBMS_RANDOM.VALUE(1002, 12000));
        offset2 := TRUNC(DBMS_RANDOM.VALUE(3000, 18900));

        IF MOD(offset1, 7) < 4 THEN
            v_gender := 'F';
        END IF;

        -- Use a single update statement with a subquery
        UPDATE customer_dim
        SET dob = (
            SELECT birth_date
            FROM DOB_data
            WHERE record_id = TRUNC(DBMS_RANDOM.VALUE(offset1,
offset2))
        ),
        gender = v_gender
        WHERE customer_key = cust_rec.customer_key;

    END LOOP;
END;
/
```

Sales Fact Table

```
INSERT INTO SalesFact
SELECT   C.DATE_KEY,
         D.MENU_KEY,
         E.PROMO_KEY,
         F.CUSTOMER_KEY,
         A.ORDERID,
         to_char(A.OrderDate,'hh24:mi'),
         UPPER(G.CompanyName),
         B.Quantity,
         B.SellingPrice AS UnitPrice,
         B.SellingPrice * (1-E.PromoPercentage) AS
Discount_Unit_Price,
         (B.SellingPrice * (1-E.PromoPercentage)) * QUANTITY AS
LineTotal
FROM NEW_ORDERS          A
JOIN NEW_ORDERDETAILS    B   ON A.ORDERID = B.ORDERID
JOIN DATE_DIM            C   ON (TRUNC(A.OrderDate) =
TRUNC(C.Cal_date))
JOIN MENU_DIM            D   ON B.menuID = D.menuID
JOIN PROMOTION_DIM       E   ON A.PromoID = E.PromoID
```

```
JOIN CUSTOMER_DIM        F   ON A.CustID = F.CustID
JOIN DeliveryCompany     G   ON A.DeliveryID = G.CompanyID
WHERE A.OrderDate between F.RowEffectiveDate AND F.RowExpirationDate;
```

2.2 Script for subsequent loading

Menu Dimension Table

```
--SUBSEQUENT LOADING(MENU_DIM)
DROP PROCEDURE Prod_Insert_Menu_Dim;
CREATE OR REPLACE PROCEDURE Prod_Insert_Menu_Dim IS
    v_null_value_count NUMBER := 0;
    v_invalid_length_count NUMBER := 0;
    v_rows_inserted NUMBER := 0;  -- Variable to store the count of
inserted rows
BEGIN
    -- Step 1: Check for NULL or empty values in critical fields
    SELECT COUNT(*) INTO v_null_value_count
    FROM menu M
    JOIN menuCategory C ON M.menuCategoryID = C.menuCategoryID
    WHERE M.menuID IS NULL
       OR TRIM(M.ItemName) IS NULL
       OR TRIM(C.CategoryName) IS NULL;


    IF v_null_value_count > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Error: NULL values found in critical
fields');
        -- Optionally log this to a table or handle as required
        RETURN;
    END IF;


    -- Step 2: Check for invalid lengths (e.g., menuName exceeding 40
characters)
    SELECT COUNT(*) INTO v_invalid_length_count
    FROM menu M
    WHERE LENGTH(TRIM(M.ItemName)) > 40;


    IF v_invalid_length_count > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Error: Invalid length in Menu Name');
        -- Optionally log this to a table or handle as required
        RETURN;
    END IF;


    -- Step 3: Insert the valid data, ensuring no duplicate entries
    INSERT INTO menu_dim (menu_key, menuID, menuName, categoryName)
```

```
    SELECT menu_dim_seq.NEXTVAL,
           M.menuID,
           UPPER(TRIM(M.ItemName)),   -- Trim and convert to
uppercase for consistency
           UPPER(TRIM(C.CategoryName))
    FROM menu M
    JOIN menuCategory C ON M.menuCategoryID = C.menuCategoryID
    WHERE NOT EXISTS (
        SELECT 1 FROM menu_dim WHERE menu_dim.menuID = M.menuID
    )
    AND LENGTH(TRIM(M.ItemName)) <= 40;  -- Ensure valid length
constraint


    -- Step 4: Capture the number of rows inserted
    v_rows_inserted := SQL%ROWCOUNT;


    -- Step 5: Log the number of rows inserted
    DBMS_OUTPUT.PUT_LINE(v_rows_inserted || ' row(s) successfully
inserted into menu_dim.');


END;
/

--check insertion
--INSERT INTO Menu VALUES (MenuID_seq.nextval,'Kopi C', '3.1',
'Strong black coffee commonly enjoyed in Malaysia.', 1);

EXEC Prod_Insert_Menu_Dim
```

Date Dimension Table

```
CREATE OR REPLACE PROCEDURE prc_subsequent_load_date_dim AS
  v_CAL_DATE DATE;
  v_FULL_DESC VARCHAR2(40);
  v_DAY_OF_WEEK NUMBER(1);
  v_MONTH_NAME  varchar(9);
  v_CAL_MONTH_YEAR NUMBER(2);
  v_CAL_YEAR_MONTH CHAR(7);
  v_CAL_QUARTER CHAR(2);
  v_CAL_YEAR_QUARTER CHAR(7);
  v_CAL_YEAR NUMBER(4);
  v_WEEKDAY_IND CHAR(1);
  new_start_date DATE;
  new_end_date DATE;
BEGIN
  -- Determine the start date as the day after the max date in
Date_dim
  SELECT MAX(cal_date) + 1 INTO new_start_date FROM Date_dim;

  -- Determine the end date as the current date
  new_end_date := SYSDATE;

  -- Ensure that there is a valid range of dates to process
  IF new_start_date > new_end_date THEN
    DBMS_OUTPUT.PUT_LINE('No new dates to load.');
    RETURN;
  END IF;

  -- Loop through each date in the range and insert into Date_dim
  WHILE (new_start_date <= new_end_date) LOOP
    v_CAL_DATE := new_start_date;
    v_FULL_DESC := TO_CHAR(new_start_date, 'DD') || ' Of ' ||
                   TO_CHAR(new_start_date, 'Month') || ' ' ||
                   TO_CHAR(new_start_date, 'Year');
    v_DAY_OF_WEEK := TO_CHAR(new_start_date, 'D');
    v_CAL_MONTH_YEAR := TO_CHAR(new_start_date, 'MM');
    v_MONTH_NAME := TO_CHAR(new_start_date, 'Month');
    v_CAL_QUARTER := 'Q' || TO_CHAR(new_start_date, 'Q');
    v_CAL_YEAR := TO_CHAR(new_start_date, 'YYYY');
    v_CAL_YEAR_MONTH := v_CAL_YEAR || '-' || v_CAL_MONTH_YEAR;
    v_CAL_YEAR_QUARTER := v_CAL_YEAR || '-' || v_CAL_QUARTER;

    -- Determine whether the date is a weekday or weekend
    IF (v_DAY_OF_WEEK BETWEEN 2 AND 6) THEN
      v_WEEKDAY_IND := 'Y';
```

```
    ELSE
      v_WEEKDAY_IND := 'N';
    END IF;

    -- Insert data into Date_dim table
    BEGIN
      INSERT INTO Date_dim (
        date_key,
        cal_date,
        full_desc,
        day_of_week,
        cal_month_year,
        cal_year_month,
        month_name,          -- Added month_name here
        cal_quarter,
        cal_year_quarter,
        cal_year,
        weekday_ind
      ) VALUES (
        date_seq.NEXTVAL,
        v_CAL_DATE,
        v_FULL_DESC,
        v_DAY_OF_WEEK,
        v_CAL_MONTH_YEAR,
        v_CAL_YEAR_MONTH,
        v_MONTH_NAME,          -- Insert month_name here
        v_CAL_QUARTER,
        v_CAL_YEAR_QUARTER,
        v_CAL_YEAR,
        v_WEEKDAY_IND
      );
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred while inserting data: '
|| SQLERRM);
        ROLLBACK;
    END;

    -- Increment the date by 1
    new_start_date := new_start_date + 1;

  END LOOP;

  DBMS_OUTPUT.PUT_LINE('Subsequent ETL process completed
successfully.');
```

```
END;
/

-- Execute the procedure
EXEC prc_subsequent_load_date_dim;
```

Promotion Dimension Table

```
DROP PROCEDURE Prod_Insert_Promotion_Dim;

-- Create or replace the procedure with advanced scrubbing features
CREATE OR REPLACE PROCEDURE Prod_Insert_Promotion_Dim IS
BEGIN
    -- Begin a transaction
    BEGIN
        -- Insert data into the promotion_dim table with advanced
scrubbing
        INSERT INTO promotion_dim (promo_key, promoID, promoName,
promoStartDate, promoEndDate, promoPercentage, promoCategory)
        SELECT promotion_dim_seq.NEXTVAL,
               p.promoID,
               UPPER(TRIM(p.promoName)),
               p.promoStartDate,
               p.promoEndDate,
               p.promoPercentage,
               UPPER(TRIM(c.CategoryName))
        FROM promotions p
        JOIN promoCategory c ON p.promoCategoryID = c.promoCategoryID
        WHERE p.promoID NOT IN (
          SELECT promoID FROM promotion_dim
        )
        AND p.promoStartDate IS NOT NULL
        AND p.promoEndDate IS NOT NULL
        AND p.promoStartDate <= p.promoEndDate
        AND p.promoPercentage >= 0
        AND TRIM(p.promoName) IS NOT NULL
        AND TRIM(c.CategoryName) IS NOT NULL;

        -- Commit the transaction
        COMMIT;

        -- Log successful insertion
        DBMS_OUTPUT.PUT_LINE('Promotion data inserted
successfully.');
```

```
    EXCEPTION
        -- Handle any errors that occur during the insert
        WHEN OTHERS THEN
            -- Rollback the transaction in case of error
            ROLLBACK;
            -- Log the error
            DBMS_OUTPUT.PUT_LINE('Error inserting promotion data: '
|| SQLERRM);
            -- Optionally, you could insert errors into an error log
table or send notifications
            -- INSERT INTO error_log (error_message, error_time)
VALUES (SQLERRM, SYSDATE);
    END;
END;
/

-- Execute the procedure
EXEC Prod_Insert_Promotion_Dim;
```

Customer Dimension Table

```
CREATE OR REPLACE PROCEDURE proc_sub_loading_cust AS
    v_rows_inserted NUMBER;
BEGIN
    -- Insert New Customers Only with Scrubbing Logic
    INSERT INTO customer_dim (customer_key, CustID, CustName,
CustCity, CustState, RowEffectiveDate, RowExpirationDate, isCurrent)
    SELECT
        cust_dim_seq.NEXTVAL,
        CUSTID,
        UPPER(NVL(TRIM(CustName), 'UNKNOWN')),
        UPPER(NVL(TRIM(CustCity), 'UNKNOWN CITY')),
        UPPER(NVL(TRIM(CustState), 'UNKNOWN STATE')),
        SYSDATE,
        TO_DATE('9999-12-31', 'yyyy-mm-dd'),
        'Y'
    FROM new_cust nc
    WHERE NOT EXISTS (
        SELECT 1
        FROM customer_dim cd
        WHERE cd.CustID = nc.CustID
        AND cd.isCurrent = 'Y'
    )
    AND CUSTID IS NOT NULL
    AND REGEXP_LIKE(CUSTID, '^\d+$')
```

```
    AND LENGTH(TRIM(CustName)) > 0
    AND LENGTH(TRIM(CustCity)) > 0
    AND LENGTH(TRIM(CustState)) > 0;

    -- Get the number of rows inserted
    v_rows_inserted := SQL%ROWCOUNT;

    -- Provide feedback based on the number of rows inserted
    IF v_rows_inserted > 0 THEN
        DBMS_OUTPUT.PUT_LINE(v_rows_inserted || ' new rows were
successfully inserted into customer_dim.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No new rows were inserted. All records
already exist or are invalid.');
    END IF;

    -- Commit the changes
    COMMIT;
END proc_sub_loading_cust;
/

INSERT INTO new_cust VALUES (customer_seq.nextval,'Choong Yam En' ,
'019-012 3456','en@gmail.com' , 'Setapak', 'Wilayah Persekutuan');
EXEC proc_sub_loading_cust;
```

Sales Fact Table

```
DROP PROCEDURE INSERT_INTO_SALESFACT;

CREATE OR REPLACE PROCEDURE INSERT_INTO_SALESFACT IS
    v_rows_inserted NUMBER := 0;
BEGIN
    -- Step 1: Insert clean and valid data into SalesFact
    INSERT INTO SalesFact
    SELECT   C.DATE_KEY,
             D.MENU_KEY,
             E.PROMO_KEY,
             F.CUSTOMER_KEY,
             A.ORDERID,
             TO_CHAR(A.OrderDate,'hh24:mi') AS OrderTime,
             UPPER(G.CompanyName) AS DeliveryCompany,
             B.Quantity,
             B.SellingPrice AS UnitPrice,
             CASE
                 WHEN E.PromoPercentage IS NOT NULL THEN
B.SellingPrice * (1-E.PromoPercentage)
```

```
                ELSE B.SellingPrice
            END AS Discount_Unit_Price,
            CASE
                WHEN E.PromoPercentage IS NOT NULL THEN
(B.SellingPrice * (1-E.PromoPercentage)) * B.Quantity
                ELSE B.SellingPrice * B.Quantity
            END AS LineTotal
    FROM NEW_ORDERS          A
    JOIN NEW_ORDERDETAILS    B   ON A.ORDERID = B.ORDERID
    JOIN DATE_DIM            C   ON TRUNC(A.OrderDate) =
TRUNC(C.Cal_date)
    JOIN MENU_DIM            D   ON B.menuID = D.menuID
    JOIN PROMOTION_DIM       E   ON A.PromoID = E.PromoID
    JOIN CUSTOMER_DIM        F   ON A.CustID = F.CustID
    JOIN DeliveryCompany     G   ON A.DeliveryID = G.CompanyID
    WHERE A.OrderDate BETWEEN F.RowEffectiveDate AND
F.RowExpirationDate
    AND A.ORDERID NOT IN (SELECT ORDERID FROM SALESFACT)   -- Avoid
duplicate orders
    AND B.Quantity > 0                                     -- Ensure
valid quantities
    AND B.SellingPrice > 0                                 -- Ensure
valid prices
    AND LENGTH(TRIM(D.menuName)) <= 40                     -- Ensure
valid menu name length
    AND LENGTH(TRIM(D.categoryName)) <= 15;                -- Ensure
valid category name length

    -- Step 2: Capture the number of rows inserted
    v_rows_inserted := SQL%ROWCOUNT;

    -- Step 3: Log the number of rows inserted
    DBMS_OUTPUT.PUT_LINE(v_rows_inserted || ' row(s) successfully
inserted into SalesFact.');

    -- Optional: Error logging and handling
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error during SalesFact insertion: ' ||
SQLERRM);
        ROLLBACK;  -- Rollback in case of errors
END;
/

EXEC INSERT_INTO_SALESFACT
```

2.3 Type 2 SCD Maintenance

2.3.1 Update the RowEffectiveDate, RowExpirationDate and IsCurrent Indicator

Customer Dimension Table

```
CREATE OR REPLACE PROCEDURE update_customer_city_with_date (
    p_CustID      IN customer_dim.CustID%TYPE,   -- Customer ID as
input
    p_newCity     IN customer_dim.CustCity%TYPE, -- New city name as
input
    p_updateDate  IN DATE                        -- Update date as
input
)
IS
    v_customer_key customer_dim.customer_key%TYPE;
BEGIN
    -- Find the current active record for the customer
    SELECT customer_key INTO v_customer_key
    FROM customer_dim
    WHERE CustID = p_CustID
      AND isCurrent = 'Y'
      FOR UPDATE;

    -- Update the current record to expire the day before the update
date
    UPDATE customer_dim
    SET RowExpirationDate = p_updateDate - 1,
        isCurrent = 'N'
    WHERE customer_key = v_customer_key;
    -- Insert a new record with the updated city and new
effective/expiration dates
    INSERT INTO customer_dim (
        customer_key,
        CustID,
        CustName,
        CustCity,
        CustState,
        DOB,
        GENDER,
        RowEffectiveDate,
        RowExpirationDate,
        isCurrent
    )
    SELECT
        cust_dim_seq.NEXTVAL,   -- New customer_key from sequence
```

```
        CustID,
        CustName,
        UPPER(p_newCity),       -- New city name in uppercase
        CustState,
        DOB,
        GENDER,
        p_updateDate,           -- New record's effective date
        TO_DATE('31/12/9999', 'DD/MM/YYYY'), -- New record's
expiration date
        'Y'                     -- Mark the new record as current
    FROM customer_dim
    WHERE customer_key = v_customer_key;

    -- Commit the changes
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Customer city updated successfully.');

EXCEPTION
    -- Handle errors, e.g., customer not found or other exceptions
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: Customer with CustID ' ||
p_CustID || ' not found or has no active record.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
        ROLLBACK;
END update_customer_city_with_date;
/
```

2.3.2 Insert New Row

```
EXEC update_customer_city_with_date(100001, 'WANGSA MAJU',
to_date('22/08/2020','dd/mm/yyyy'))

EXEC update_customer_city_with_date(100002, 'CHERAS',
to_date('23/09/2020','dd/mm/yyyy'))

EXEC update_customer_city_with_date(100003, 'AMPANG',
to_date('24/10/2020','dd/mm/yyyy'))
```

Chapter 3 : Business Analytics Reports

3.1 Yam Jason

3.1.1 Sales Trend by Category and Year and Projections

SQL Code:

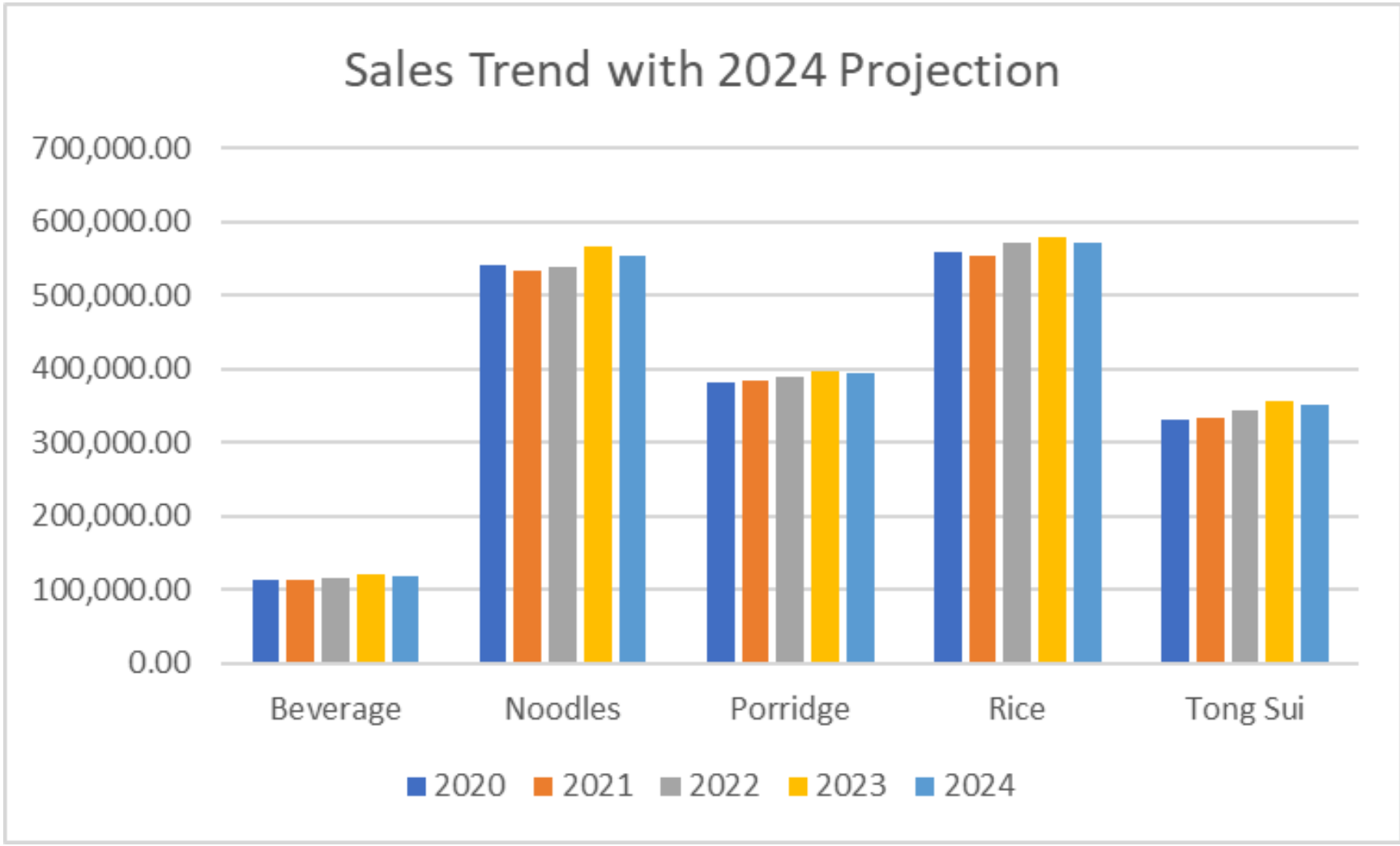
```
SET PAGESIZE 33
SET LINESIZE 115

-- Prompt the user for the years
ACCEPT startYear PROMPT 'Enter the start year: '
ACCEPT endYear PROMPT 'Enter the end year: '

-- Calculate the next year
COLUMN nextYear NEW_VALUE nextYear NOPRINT;
SELECT TO_CHAR(&endYear + 1) AS nextYear FROM dual;

TTITLE ON
BTITLE ON

TTITLE CENTER
'============================================================' SKIP
1 -
       CENTER 'Sales Trend by Category and Year
(&startYear-&endYear)' SKIP 1 -
       CENTER '   and Projection for Year &nextYear' SKIP 1 -
       CENTER
'============================================================' SKIP
1 -
       LEFT 'Date Generated: ' _DATE -
       RIGHT 'Page ' SQL.PNO

BTITLE CENTER '-----End of Query-----'

COLUMN "CATEGORYNAME" HEADING 'CATEGORY NAME';
COLUMN "TOTAL_SALES" FORMAT $9999,999.99 HEADING 'TOTAL SALES';
COLUMN "PREVIOUS_YEAR_SALES" FORMAT $9999,999.99 HEADING
'PREVIOUS YEAR SALES';
COLUMN "SALES_CHANGE" FORMAT $9999,999.99 HEADING 'SALES
CHANGE';
COLUMN "SALES_CHANGE_PERC" HEADING 'SALES CHANGE (%)';
COLUMN "Projected_Sales_For_Next_Year" FORMAT $9999,999.99
HEADING '&nextYear T_Sales Projection';
```

```
BREAK ON CategoryName SKIP 1 ON Projected_Sales_For_Next_Year

WITH SalesByCategory AS (
    SELECT
        MD.CategoryName,
        DD.cal_year AS Year,
        SUM(SF.LineTotal) AS Total_Sales
    FROM
        SalesFact SF
    JOIN
        Date_dim DD ON SF.DATE_KEY = DD.date_key
    JOIN
        menu_dim MD ON SF.MENU_KEY = MD.MENU_KEY
    WHERE DD.cal_year BETWEEN &startYear AND &endYear  -- Filter
for years
    GROUP BY
        MD.CategoryName, DD.cal_year
),
SalesWithLag AS (
    SELECT
        CategoryName,
        Year,
        Total_Sales,
        LAG(Total_Sales, 1, 0) OVER (PARTITION BY CategoryName
ORDER BY Year) AS Previous_Year_Sales,
        (Total_Sales - LAG(Total_Sales, 1, 0) OVER (PARTITION BY
CategoryName ORDER BY Year)) AS Sales_Change,
        CASE
            WHEN LAG(Total_Sales, 1, 0) OVER (PARTITION BY
CategoryName ORDER BY Year) = 0 THEN NULL
            ELSE ROUND((Total_Sales - LAG(Total_Sales, 1, 0)
OVER (PARTITION BY CategoryName ORDER BY Year)) /
                       LAG(Total_Sales, 1, 0) OVER (PARTITION
BY CategoryName ORDER BY Year) * 100, 2)
        END AS Sales_Change_Perc,
        AVG(Total_Sales) OVER (PARTITION BY CategoryName) AS
AVGTotalSales
    FROM SalesByCategory
),
ProjectedSales AS (
    SELECT
        CategoryName,
        SUM(Sales_Change_Perc) / COUNT(Sales_Change_Perc) AS
AvgGrowthRate
```

```
    FROM SalesWithLag
    WHERE Sales_Change_Perc IS NOT NULL
    GROUP BY CategoryName
)
SELECT
    SWL.CategoryName,
    SWL.Year,
    SWL.Total_Sales,
    SWL.Previous_Year_Sales,
    SWL.Sales_Change,
    SWL.Sales_Change_Perc,
    ROUND(SWL.AVGTotalSales * (1 + (PS.AvgGrowthRate / 100)), 2)
AS Projected_Sales_For_Next_Year
FROM
    SalesWithLag SWL
JOIN
    ProjectedSales PS ON SWL.CategoryName = PS.CategoryName
ORDER BY
    SWL.CategoryName, SWL.Year;

CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF
BTITLE OFF
```

Output:

```
Enter the start year: 2020
Enter the end year: 2023
old   1: SELECT TO_CHAR(&endYear + 1) AS nextYear FROM dual
new   1: SELECT TO_CHAR(2023 + 1) AS nextYear FROM dual
old  12:     WHERE DD.cal_year BETWEEN &startYear AND &endYear  -- Filter for years
new  12:     WHERE DD.cal_year BETWEEN 2020 AND 2023  -- Filter for years
                            ========================================================
                                 Sales Trend by Category and Year (2020-2023)
                                           and Projection for Year 2024
                            ========================================================
Date Generated: 20-SEP-24                                                                  Page          1
CATEGORY NAME         YEAR   TOTAL SALES PREVIOUS YEAR SALES  SALES CHANGE SALES CHANGE (%) 2024 T_Sales Projection
--------------- ---------- ------------ -------------------- ------------- ---------------- ------------------------
BEVERAGE              2020   $113,490.21                $.00   $113,490.21                            $118,948.89
                      2021   $113,139.76         $113,490.21      -$350.45             -.31
                      2022   $116,515.06         $113,139.76     $3,375.30             2.98
                      2023   $121,665.50         $116,515.06     $5,150.44             4.42

NOODLES               2020   $541,124.95                $.00   $541,124.95                            $553,740.92
                      2021   $533,574.50         $541,124.95    -$7,550.45             -1.4
                      2022   $539,692.65         $533,574.50     $6,118.15             1.15
                      2023   $566,405.75         $539,692.65    $26,713.10             4.95

PORRIDGE              2020   $382,701.62                $.00   $382,701.62                            $393,057.70
                      2021   $383,616.10         $382,701.62       $914.48              .24
                      2022   $390,045.41         $383,616.10     $6,429.31             1.68
                      2023   $396,866.51         $390,045.41     $6,821.10             1.75

RICE                  2020   $558,590.70                $.00   $558,590.70                            $572,428.12
                      2021   $554,854.20         $558,590.70    -$3,736.50             -.67
                      2022   $571,514.55         $554,854.20    $16,660.35                3
                      2023   $578,273.20         $571,514.55     $6,758.65             1.18

TONG SUI              2020   $329,931.51                $.00   $329,931.51                            $350,187.41
                      2021   $332,977.41         $329,931.51     $3,045.90              .92
                      2022   $343,538.62         $332,977.41    $10,561.21             3.17
                      2023   $357,387.63         $343,538.62    $13,849.01             4.03
                                                -----End of Query-----
20 rows selected.
```


Sales Trend with 2024 Projection
700,000.00
600,000.00
500,000.00
400,000.00
300,000.00
200,000.00
100,000.00
0.00
Beverage
Noodles
Porridge
Rice
Tong Sui
2020
2021
2022
2023
2024

The query and the bar chart provided valuable insights into the sales trend of the company across various categories from 2020 to 2023, along with projections for 2024. Notably, the projections indicate that total sales across all categories are expected to decline compared to the previous year, suggesting a potential slowdown in growth. These insights are crucial for decision-makers to assess the company's performance and shape future strategies. For instance, implementing targeted marketing campaigns or launching promotions specifically for categories with significant declines could help revitalize sales. Additionally, this situation presents opportunities for product innovation and competitive pricing strategies to strengthen underperforming categories. The formula to calculate the projection is (averageTotalSales) * (1+ (avgGrowthRate /100)).

3.1.2 Promotion Effectiveness Analysis

SQL Code:

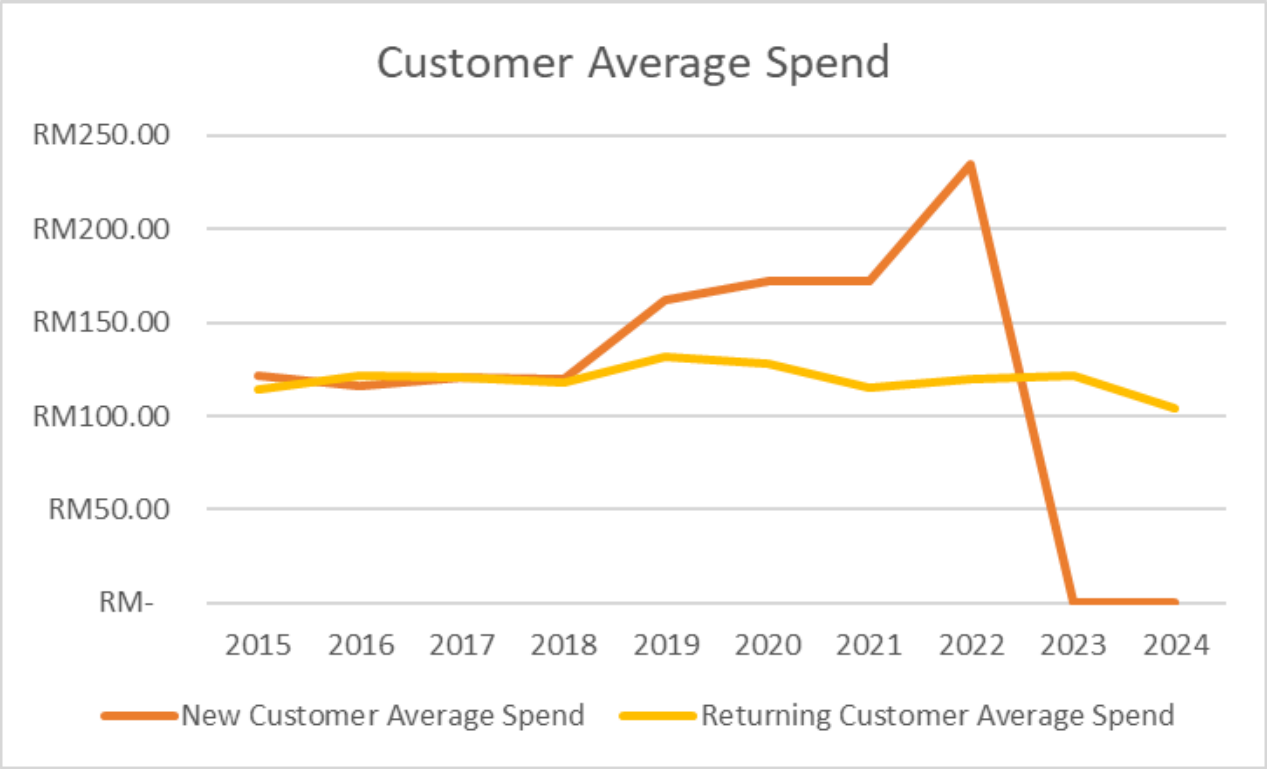
```
set pagesize 19
set linesize 98

-- Prompt the user for the category name
ACCEPT promo PROMPT 'Enter the Promotion: '

TTITLE ON
BTITLE ON

TTITLE CENTER '==================================' SKIP 1 -
       CENTER 'Promotion Effectiveness Analysis' SKIP 1 -
       CENTER '==================================' SKIP 1 -
       LEFT 'Date Generated: ' _DATE -
       RIGHT 'Page ' SQL.PNO

BTITLE CENTER '-----End of Query-----'

 COLUMN "PROMONAME" HEADING 'PROMOTION NAME' FORMAT A27;
 COLUMN "NC_TOTAL_CUSTOMERS" HEADING 'TOTAL NEW|CUST';
 COLUMN "RC_TOTAL_CUSTOMERS" HEADING 'TOTAL RETURNING|CUST';
 COLUMN "NC_TOTAL_REVENUE" HEADING 'NC TOTAL|REVENUE' FORMAT
$999999.99;
 COLUMN "RC_TOTAL_REVENUE" HEADING 'RC TOTAL|REVENUE' FORMAT
$999999.99;
 COLUMN "NC_AVG_SPEND" HEADING 'NC AVG|SPEND' FORMAT $999.99;
 COLUMN "RC_AVG_SPEND" HEADING 'RC AVG|SPEND' FORMAT $999.99;


WITH CustomerFirstPurchase AS (
    -- Find the first date each customer made a purchase
    SELECT
        CUSTOMER_KEY,
        MIN(DD.cal_year) AS first_purchase_year
    FROM
        SalesFact SF
    JOIN
        Date_dim DD ON SF.DATE_KEY = DD.date_key
    GROUP BY
        CUSTOMER_KEY
),
CustomerOrders AS (
```

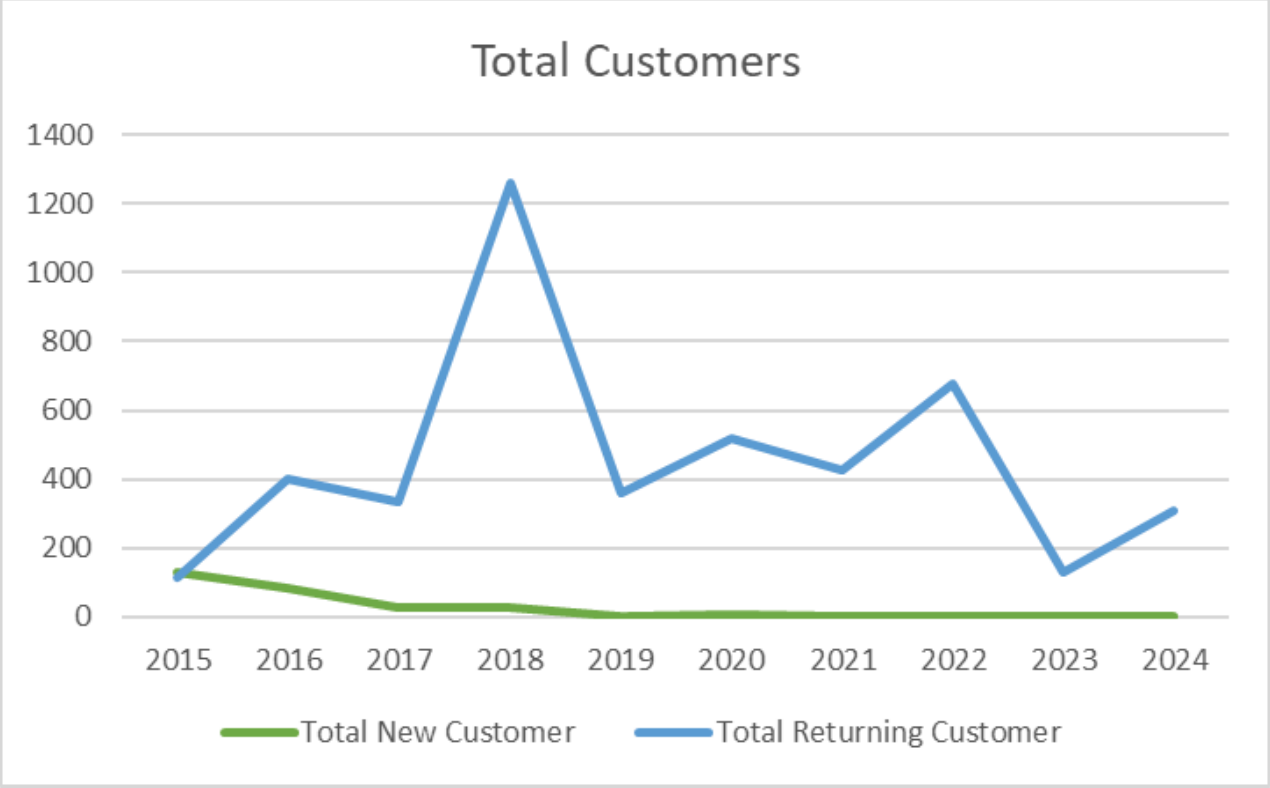
```
    -- Identify both new and returning customer orders
    SELECT
        SF.CUSTOMER_KEY,
        SF.PROMO_KEY,
        CASE
            WHEN CFP.first_purchase_year = DD.cal_year THEN 'New
Customer'
            ELSE 'Returning Customer'
        END AS customer_type,
        SUM(SF.LineTotal) AS Total_Spent
    FROM
        SalesFact SF
    JOIN
        Date_dim DD ON SF.DATE_KEY = DD.date_key
    JOIN
        CustomerFirstPurchase CFP ON SF.CUSTOMER_KEY =
CFP.CUSTOMER_KEY
    GROUP BY
        SF.CUSTOMER_KEY, SF.PROMO_KEY, CFP.first_purchase_year,
DD.cal_year
)
-- Pivot the result to show New Customer and Returning Customer
as columns
SELECT *
FROM (
    SELECT
        PD.PromoName,
        CO.customer_type,
        COUNT(DISTINCT CO.CUSTOMER_KEY) AS Total_Customers,
        SUM(CO.Total_Spent) AS Total_Revenue,
        AVG(CO.Total_Spent) AS AVG_Spend
    FROM
        CustomerOrders CO
    JOIN
        PROMOTION_DIM PD ON CO.PROMO_KEY = PD.PROMO_KEY
    WHERE
        PD.PromoName LIKE UPPER('%&promo%')
    GROUP BY
        PD.PromoName, CO.customer_type
)
PIVOT (
    SUM(Total_Customers) AS Total_Customers, SUM(Total_Revenue)
AS Total_Revenue
    , AVG(AVG_Spend) AS AVG_Spend
```

```
    FOR customer_type IN ('New Customer' AS "NC", 'Returning
Customer' AS "RC")
)
ORDER BY PromoName;


CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF
BTITLE OFF
```

Output:

```
Enter the Promotion: chinese
old  46:          PD.PromoName LIKE UPPER('%&promo%')
new  46:          PD.PromoName LIKE UPPER('%chinese%')
                                =================================
                                 Promotion Effectiveness Analysis
                                =================================
Date Generated: 20-SEP-24                                                                  Page           1
                              TOTAL NEW    NC TOTAL      NC AVG TOTAL RETURNING    RC TOTAL    RC AVG
PROMOTION NAME                     CUST     REVENUE       SPEND             CUST     REVENUE     SPEND
--------------------------- ---------- ----------- ----------- --------------- ----------- --------
CHINESE NEW YEAR SALE 2015         127   $15405.12     $121.30             116    $13213.6  $113.91
CHINESE NEW YEAR SALE 2016          85    $9835.12     $115.71             399   $48625.36  $121.87
CHINESE NEW YEAR SALE 2017          25    $3022.48     $120.90             334   $40218.24  $120.41
CHINESE NEW YEAR SALE 2018          28     $350.65     $119.67            1260  $148125.12  $117.56
CHINESE NEW YEAR SALE 2019           3      $485.1     $161.70             360   $47330.28  $131.47
CHINESE NEW YEAR SALE 2020           6     $1030.2     $171.70             519   $66212.81  $127.58
CHINESE NEW YEAR SALE 2021           1      $172.5     $172.50             428    $49195.5  $114.94
CHINESE NEW YEAR SALE 2022           1      $234.8     $243.80             675       $80702  $119.56
CHINESE NEW YEAR SALE 2023                                                 127   $15384.72  $121.14
CHINESE NEW YEAR SALE 2024                                                 308   $31952.06  $103.74
                                      -----End of Query-----

10 rows selected.
```


Customer Average Spend
RM250.00
RM200.00
RM150.00
RM100.00
RM50.00
RM-
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
New Customer Average Spend
Returning Customer Average Spend


Total Customers
1400
1200
1000
800
600
400
200
0
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
Total New Customer
Total Returning Customer

The query and the line charts show the descriptive analysis of promotion effectiveness by including the count of new customers and returning customers and also the average spend. In this query, Chinese New Year is entered as an input to view the effectiveness of this promotion. This analysis highlights that Chinese New Year promotions from 2015 to 2024 have been more successful in retaining returning customers than attracting new ones but the average spend of the new customers seems to be higher compared to returning customers. By this observation, the decision makers will be able to see that the promotion worked pretty well but the number of new customers that used it is low, indicating a sign to work on customer acquisition. This can allow the decision makers to increase the budget of marketing, by advertising the promotions more effectively to spread awareness to attract more new customers.

3.1.3 Customer Segmentation Based on Category

SQL Code:

```
SET PAGESIZE 30
SET LINESIZE 103
-- Prompt the user for the category name
ACCEPT categoryName PROMPT 'Enter the Category (TONG
SUI/RICE/NOODLES/BEVERAGE/PORRIDGE): '
-- Set up titles
TTITLE CENTER
'================================================================
===' SKIP 1 -
       CENTER 'Customer Segmentation Based on &categoryName'
SKIP 1 -
       CENTER 'Low: T.Spent <100, Medium: T.Spent 100-200, High:
T.Spent >200' SKIP 1 -
       CENTER
'================================================================
===' SKIP 1 -
       LEFT 'Date Generated: ' _DATE -
       RIGHT 'Page ' SQL.PNO

BTITLE CENTER '-----End of Query-----'
COLUMN "AGE_RANGE" HEADING 'AGE';
COLUMN "TOTAL_SPEND" HEADING 'TOTAL|SPEND' FORMAT $999999.99;
COLUMN "AVERAGE_SPEND" HEADING 'AVERAGE|SPEND' FORMAT $9999.99;
COLUMN "Low Spenders Count" HEADING 'LOW|SPENDERS|COUNT';
COLUMN "Medium Spenders Count" HEADING 'MEDIUM|SPENDERS|COUNT';
COLUMN "High Spenders Count" HEADING 'HIGH|SPENDERS|COUNT';
COLUMN "LOW SPENDERS %" HEADING 'LOW|SPENDERS|%' FORMAT 99.99;
COLUMN "MEDIUM SPENDERS %" HEADING 'MEDIUM|SPENDERS|%' FORMAT
99.99;
COLUMN "HIGH SPENDERS %" HEADING 'HIGH|SPENDERS|%' FORMAT 99.99;

-- Break on SalesYear and Age_Range
BREAK ON SalesYear SKIP 1

-- Compute total for Number_of_Customers and Total_Spend on each
Age_Range
COMPUTE SUM OF "Low Spenders Count" ON SalesYear
COMPUTE SUM OF "Medium Spenders Count" ON SalesYear
COMPUTE SUM OF "High Spenders Count" ON SalesYear
COMPUTE SUM OF Total_Spend ON SalesYear

WITH CustomerDemographics AS (
    SELECT
```

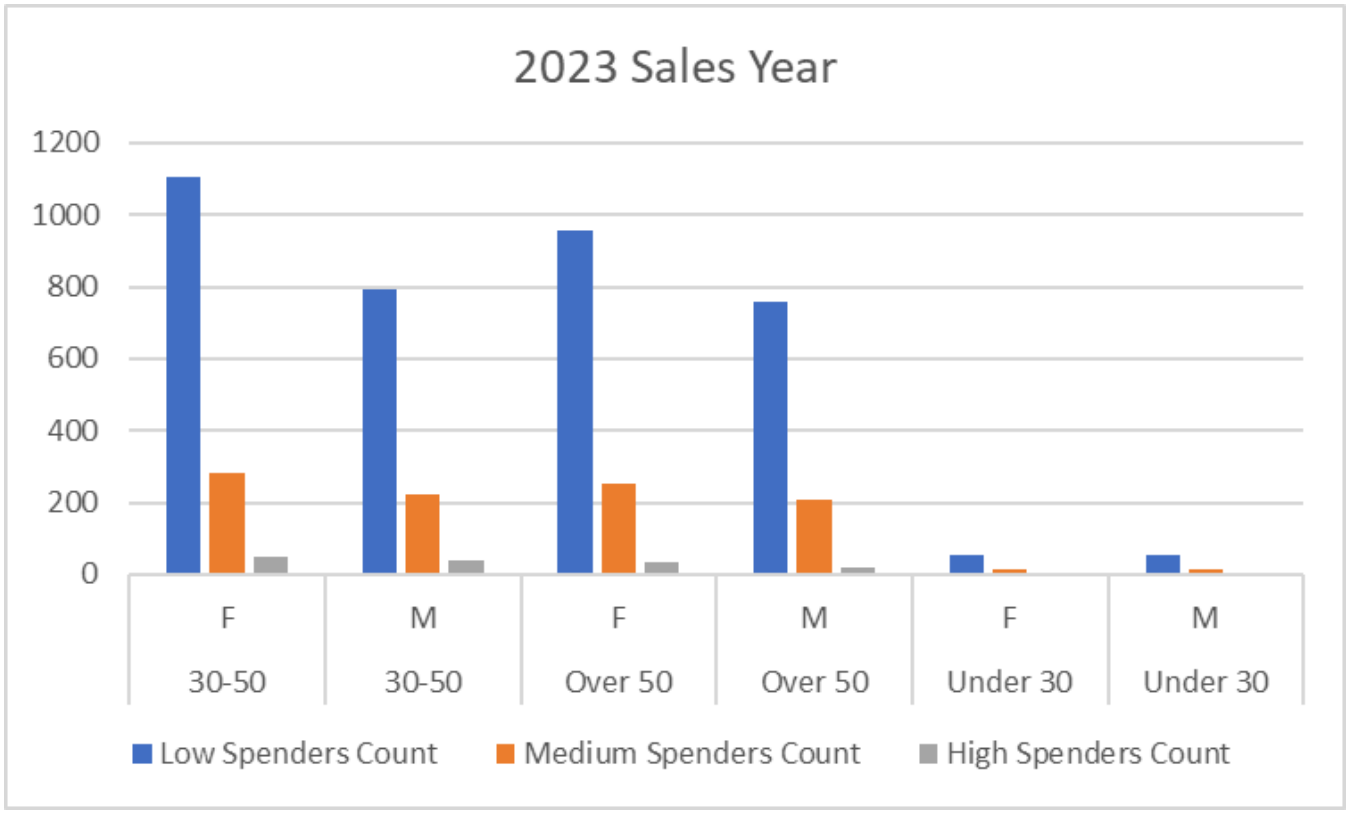
```
        CUSTOMER_KEY,
        TRUNC(months_between(sysdate, dob) / 12) AS Age,  --
Calculating age
        Gender
    FROM Customer_dim
),
CustomerSpending AS (
    SELECT
        SF.CUSTOMER_KEY,
        MD.CategoryName,
        DD.Cal_Year AS SalesYear,
        SUM(SF.LineTotal) AS Total_Spent
    FROM SalesFact SF
    JOIN MENU_DIM MD ON SF.MENU_KEY = MD.MENU_KEY
    JOIN Date_Dim DD ON SF.DATE_KEY = DD.DATE_KEY  -- Join with
Date_Dim for date information
    WHERE MD.CATEGORYNAME = UPPER('&categoryName')  -- Use user
input
    AND DD.Cal_Year IN (2022, 2023)  -- Filter for 2022 and 2023
    GROUP BY SF.CUSTOMER_KEY, MD.CategoryName, DD.Cal_Year
),
SegmentedCustomers AS (
    SELECT
        CD.CUSTOMER_KEY,
        SalesYear,
        CASE
            WHEN CD.Age < 30 THEN 'Under 30'
            WHEN CD.Age BETWEEN 30 AND 50 THEN '30-50'
            ELSE 'Over 50'
        END AS Age_Range,
        CD.Gender,
        CS.CategoryName,
        CS.Total_Spent,
        CASE
            WHEN CS.Total_Spent < 100 THEN 'Low Spender'
            WHEN CS.Total_Spent BETWEEN 100 AND 200 THEN 'Medium
Spender'
            ELSE 'High Spender'
        END AS Spending_Category
    FROM CustomerDemographics CD
    JOIN CustomerSpending CS ON CD.CUSTOMER_KEY =
CS.CUSTOMER_KEY
)
SELECT
    SalesYear,
```

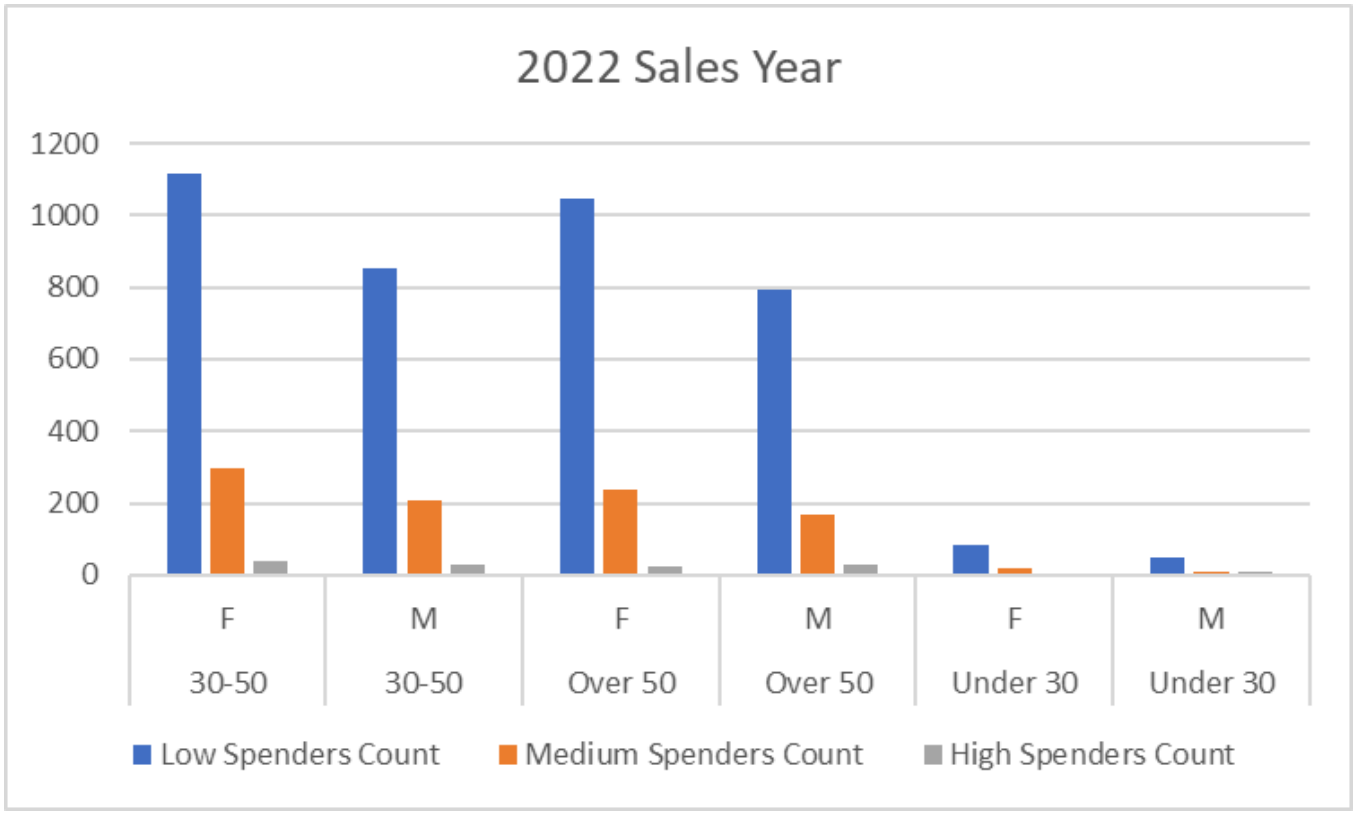
```
    Age_Range,
    Gender,
    AVG(Average_Spend) AS Average_Spend,
    SUM(Total_Spend) AS Total_Spend,
    SUM("Low Spenders") AS "Low Spenders Count",
    SUM("Medium Spenders") AS "Medium Spenders Count",
    SUM("High Spenders") AS "High Spenders Count",
    -- Calculating percentage for each spender category
    ROUND( (SUM("Low Spenders") / (SUM("Low Spenders") +
SUM("Medium Spenders") + SUM("High Spenders"))) * 100, 2) AS
"LOW SPENDERS %",
    ROUND( (SUM("Medium Spenders") / (SUM("Low Spenders") +
SUM("Medium Spenders") + SUM("High Spenders"))) * 100, 2) AS
"MEDIUM SPENDERS %",
    ROUND( (SUM("High Spenders") / (SUM("Low Spenders") +
SUM("Medium Spenders") + SUM("High Spenders"))) * 100, 2) AS
"HIGH SPENDERS %"
FROM (
    SELECT
        SalesYear,
        Age_Range,
        Gender,
        Spending_Category,
        COUNT(CUSTOMER_KEY) AS Number_of_Customers,
        AVG(Total_Spent) AS Average_Spend,
        SUM(Total_Spent) AS Total_Spend
    FROM SegmentedCustomers
    GROUP BY
        SalesYear,
        Age_Range,
        Gender,
        Spending_Category
)
PIVOT (
    SUM(Number_of_Customers) FOR Spending_Category IN ('Low
Spender' AS "Low Spenders", 'Medium Spender' AS "Medium
Spenders", 'High Spender' AS "High Spenders")
)
GROUP BY SalesYear, Age_Range, Gender
ORDER BY SalesYear, Age_Range, Gender;
-- Reset after report
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF
```

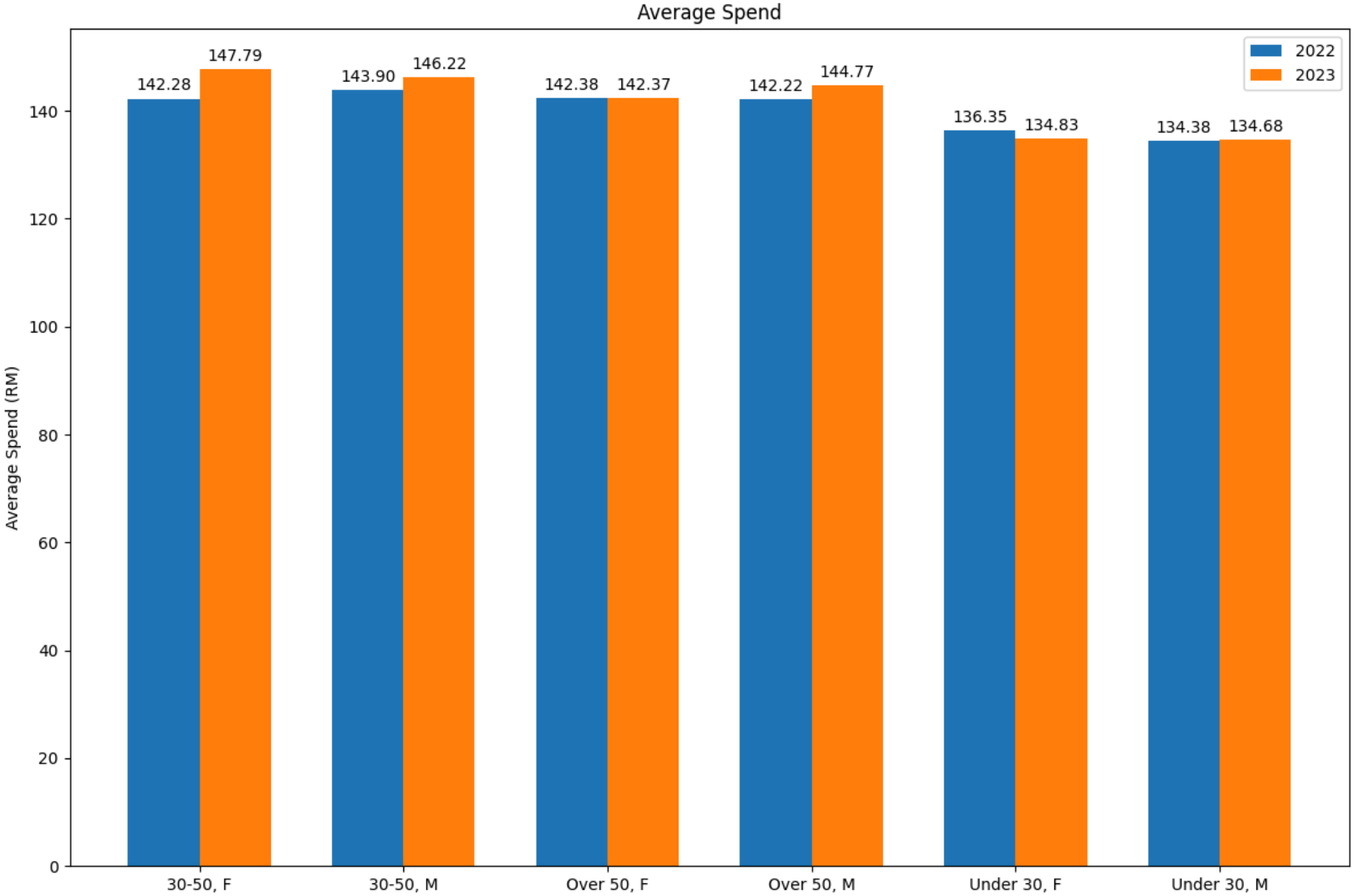
```
BTITLE OFF
```

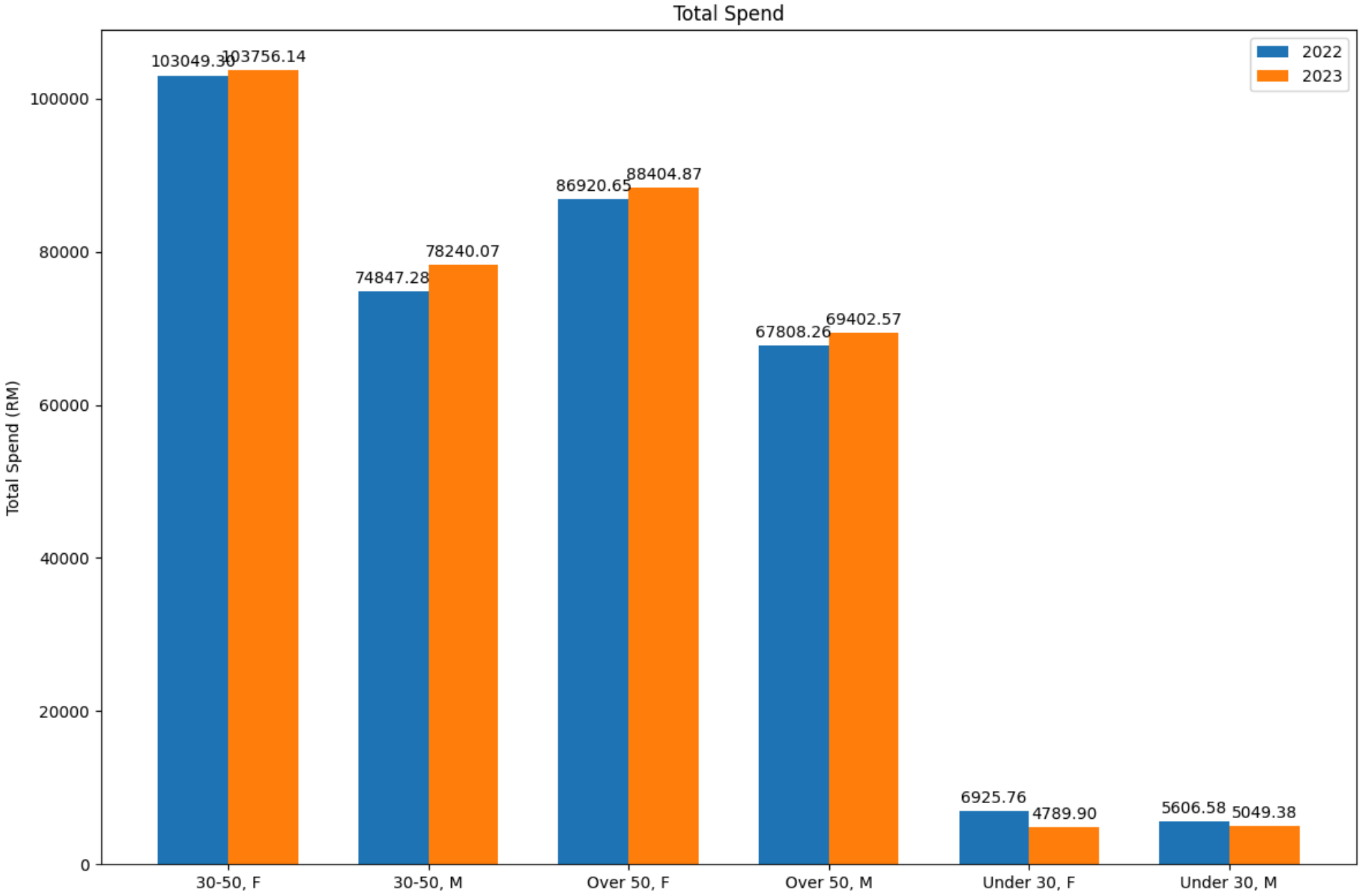
Output:

```
Enter the Category (TONG SUI/RICE/NOODLES/BEVERAGE/PORRIDGE): TONG SUI
old  17:     WHERE MD.CATEGORYNAME = UPPER('&categoryName')  -- Use user input
new  17:     WHERE MD.CATEGORYNAME = UPPER('TONG SUI')  -- Use user input

                    ======================================================================
                                  Customer Segmentation Based on TONG SUI
                      Low: T.Spent <100, Medium: T.Spent 100-200, High: T.Spent >200
                    ======================================================================
Date Generated: 20-SEP-24                                                          Page           1
                                                    LOW      MEDIUM       HIGH        LOW     MEDIUM       HIGH
                          AVERAGE       TOTAL   SPENDERS   SPENDERS   SPENDERS   SPENDERS   SPENDERS   SPENDERS
 SALESYEAR AGE      G     SPEND       SPEND      COUNT      COUNT      COUNT          %          %          %
---------- -------- - --------- ----------- ---------- ---------- ---------- ---------- ---------- ----------
      2022 30-50    F   $142.28  $103049.30       1115        298         39      76.79      20.52       2.69
           30-50    M   $143.90   $74728.48        853        208         30      78.18      19.07       2.75
           Over 50  F   $142.38   $86920.65       1047        237         25      79.98      18.11       1.91
           Over 50  M   $142.22   $67927.06        795        169         30      79.98      17.00       3.02
           Under 30 F   $136.35    $6925.76         85         17          2      81.73      16.35       1.92
           Under 30 M   $134.38    $5606.58         49         11          8      72.06      16.18      11.76
**********                      ----------- ---------- ---------- ----------
sum                      $345157.83       3944        940        134

      2023 30-50    F   $147.79  $103756.13       1106        280         51      76.97      19.49       3.55
           30-50    M   $146.22   $77997.97        791        221         38      75.33      21.05       3.62
           Over 50  F   $142.37   $88404.87        959        253         34      76.97      20.30       2.73
           Over 50  M   $144.77   $69644.67        763        210         21      76.76      21.13       2.11
           Under 30 F   $134.83    $4789.90         53         13          2      77.94      19.12       2.94
           Under 30 M   $134.68    $5049.38         56         13          2      78.87      18.31       2.82
**********                      ----------- ---------- ---------- ----------
sum                      $349642.93       3728        990        148

                                        -----End of Query-----

12 rows
```


2023 Sales Year
1200
1000
800
600
400
200
0
F
30-50
M
30-50
F
Over 50
M
Over 50
F
Under 30
M
Under 30
Low Spenders Count
Medium Spenders Count
High Spenders Count


2022 Sales Year
1200
1000
800
600
400
200
0
F
30-50
M
30-50
F
Over 50
M
Over 50
F
Under 30
M
Under 30
Low Spenders Count
Medium Spenders Count
High Spenders Count


Average Spend
2022
2023
Average Spend (RM)
0
20
40
60
80
100
120
140
142.28
147.79
143.90
146.22
142.38
142.37
142.22
144.77
136.35
134.83
134.38
134.68
30-50, F
30-50, M
Over 50, F
Over 50, M
Under 30, F
Under 30, M


Total Spend
2022
2023
Total Spend (RM)
0
20000
40000
60000
80000
100000
103049.30
103756.14
74847.28
78240.07
86920.65
88404.87
67808.26
69402.57
6925.76
4789.90
5606.58
5049.38
30-50, F
30-50, M
Over 50, F
Over 50, M
Under 30, F
Under 30, M

For this particular query, it is about customer segmentation based on a category input from the user. This query allows the decision makers to know how different segments of people contribute to the sales of a certain category and which segment they have to focus on improving. Based on the query and the graphs, it can be seen that the majority of the spenders from the category entered (Tong Sui) are low spenders and belong to the 30-50 and above 50 age range and the female gender dominates the male gender in spending too. With this information, the decision makers could come up with new products that are low priced, more preferred by female customers, and more health-conscious that can tailor to the desire of the 30-50 and above 50 age range. This decision may increase their sales significantly if implemented right. For the worst performing age range, which is the young generation below 30 years old, the decision makers could innovate new products that would align with the trends and preferences of the young generation. For example, develop aesthetically pleasing products or unique presentation styles that encourage young customers to share their purchases on social media, increasing the company's visibility.

3.2 Wong Yee En

3.2.1 Year-over-year Quarterly Sales Growth Analysis (2019-2023) and Projection (2024)

SQL Code:

```
-- Set the format for the report
SET linesize 110
SET pagesize 40

-- Accept the number of previous years for analysis
ACCEPT NumYears PROMPT 'Enter the number of previous years you
want to analyze: '

-- Define the current year and the start year based on user
input
COLUMN PreviousYear NEW_VALUE PreviousYear NOPRINT;
COLUMN StartYear NEW_VALUE StartYear NOPRINT;

-- Get the current year and calculate the start year
SELECT TO_NUMBER(TO_CHAR(SYSDATE, 'YYYY')) - 1 AS PreviousYear
FROM dual;
SELECT &PreviousYear - &NumYears + 1 AS StartYear FROM dual;

-- Column formatting
COLUMN CAL_YEAR FORMAT 9999
COLUMN Quarter FORMAT A8 HEADING 'QUARTER';
COLUMN Year1 FORMAT 9999 HEADING 'YEAR1';
COLUMN SalesYear1 FORMAT $9999,999.99 HEADING 'SALESYEAR1';
COLUMN SalesYear2 FORMAT $9999,999.99 HEADING 'SALESYEAR2';
COLUMN AbsoluteGrowth FORMAT $9999,999.99 HEADING
'Absolute|Growth';
COLUMN GrowthPercentage FORMAT 999.99 HEADING 'Growth %';
COLUMN ProjectedSales2024 FORMAT $9999,999.99 HEADING
'Projected|Sales 2024';
COLUMN ProjectedGrowth2024 FORMAT 999.99 HEADING
'Projected|Growth 2024 %';

-- Create or replace the view with a filter for the dynamic
year range
CREATE OR REPLACE VIEW Quarter_SALES AS
SELECT
    D.cal_year,
    D.cal_quarter,
    SUM(SF.linetotal) AS TotalQuarterSales,
    COUNT(SF.orderid) AS TotalTransactions
```

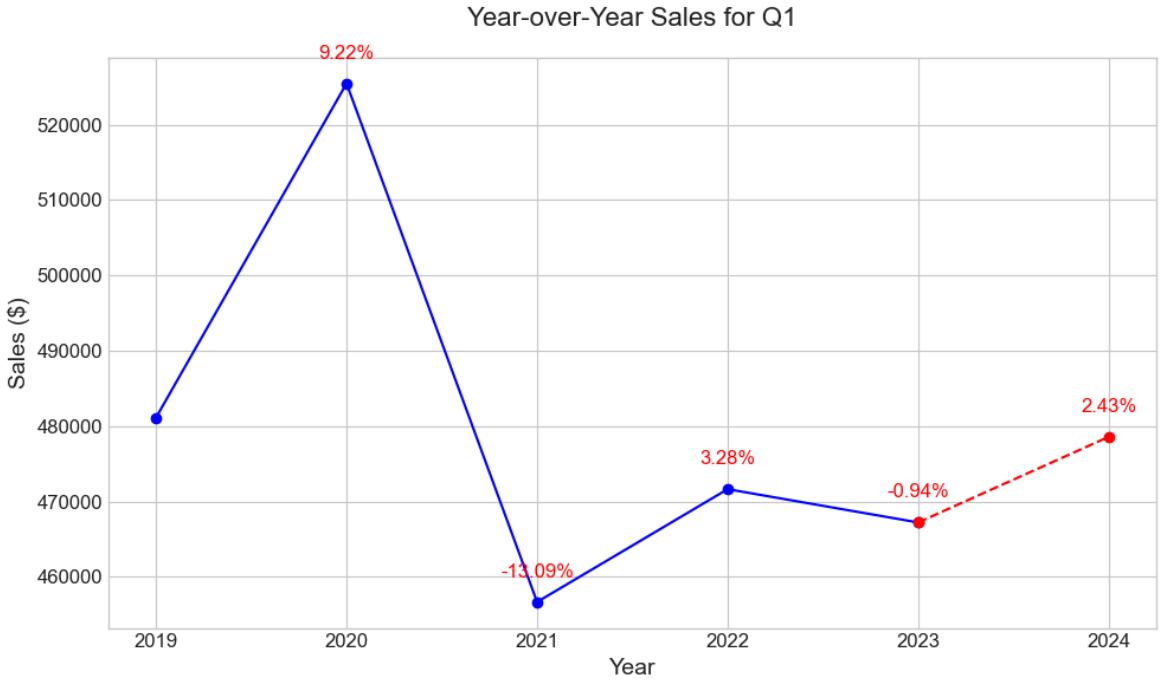
```
FROM salesfact SF
JOIN Date_dim D ON SF.date_key = D.date_key
WHERE D.cal_year BETWEEN &StartYear AND &PreviousYear
GROUP BY D.cal_year, D.cal_quarter;

-- Query for Year-over-Year analysis based on user input
TTITLE ON
TTITLE CENTER
'================================================================
======' SKIP 1 -
       CENTER 'Year-over-Year Quarterly Sales Growth Analysis
And Projection (2024)' SKIP 1 -
       CENTER
'================================================================
======' SKIP 1 LEFT 'DATE: ' _DATE SKIP 1 LEFT  'PAGE: ' FORMAT
999 SQL.PNO SKIP 1

-- Break on Quarter, AVGSalesPerQuarter and compute the sum of
SalesYear1
BREAK ON Quarter SKIP 1 ON ProjectedSales2024 ON
Growth2023to2024
COMPUTE AVG LABEL 'Average: ' OF SalesYear1 GrowthPercentage
AbsoluteGrowth ON Quarter

WITH HistoricalData AS (
    SELECT
        Q.cal_quarter AS Quarter,
        Q.cal_year AS Year1,
        Q.TotalQuarterSales AS SalesYear1,
        LAG(Q.TotalQuarterSales, 1) OVER (PARTITION BY
Q.cal_quarter ORDER BY Q.cal_year) AS SalesYear2,
        Q.TotalQuarterSales - LAG(Q.TotalQuarterSales, 1) OVER
(PARTITION BY Q.cal_quarter ORDER BY Q.cal_year) AS
AbsoluteGrowth,
        ((Q.TotalQuarterSales - LAG(Q.TotalQuarterSales, 1)
OVER (PARTITION BY Q.cal_quarter ORDER BY Q.cal_year))
        / NULLIF(LAG(Q.TotalQuarterSales, 1) OVER (PARTITION BY
Q.cal_quarter ORDER BY Q.cal_year), 0) * 100) AS
GrowthPercentage,
        AVG(Q.TotalQuarterSales) OVER (PARTITION BY
Q.cal_quarter) AS AVGSalesPerQuarter
    FROM Quarter_SALES Q
    WHERE Q.cal_year BETWEEN &StartYear AND &PreviousYear
)
```

```
-- Calculate Average Growth Rate for Each Quarter and growth
from 2023 to 2024
, GrowthRates AS (
    SELECT
        Quarter,
        AVG(GrowthPercentage) AS AvgGrowthRate
    FROM HistoricalData
    WHERE GrowthPercentage IS NOT NULL
    GROUP BY Quarter
)
-- Project Future Sales for 2024 and calculate growth
percentage from 2023 to 2024
SELECT
    H.Quarter,
    H.Year1,
    H.SalesYear1,
    H.SalesYear2,
    H.AbsoluteGrowth,
    H.GrowthPercentage,
    H.AVGSalesPerQuarter * (1 + G.AvgGrowthRate / 100) AS
ProjectedSales2024,
    CASE
        WHEN H.Year1 = &PreviousYear THEN
            ROUND((H.AVGSalesPerQuarter * (1 + G.AvgGrowthRate
/ 100) - H.SalesYear1) / NULLIF(H.SalesYear1, 0) * 100,2)
        ELSE
            NULL
    END AS ProjectedGrowth2024
FROM HistoricalData H
JOIN GrowthRates G ON H.Quarter = G.Quarter
ORDER BY H.Quarter, H.Year1;

CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTE
TTITLE OFF;
```

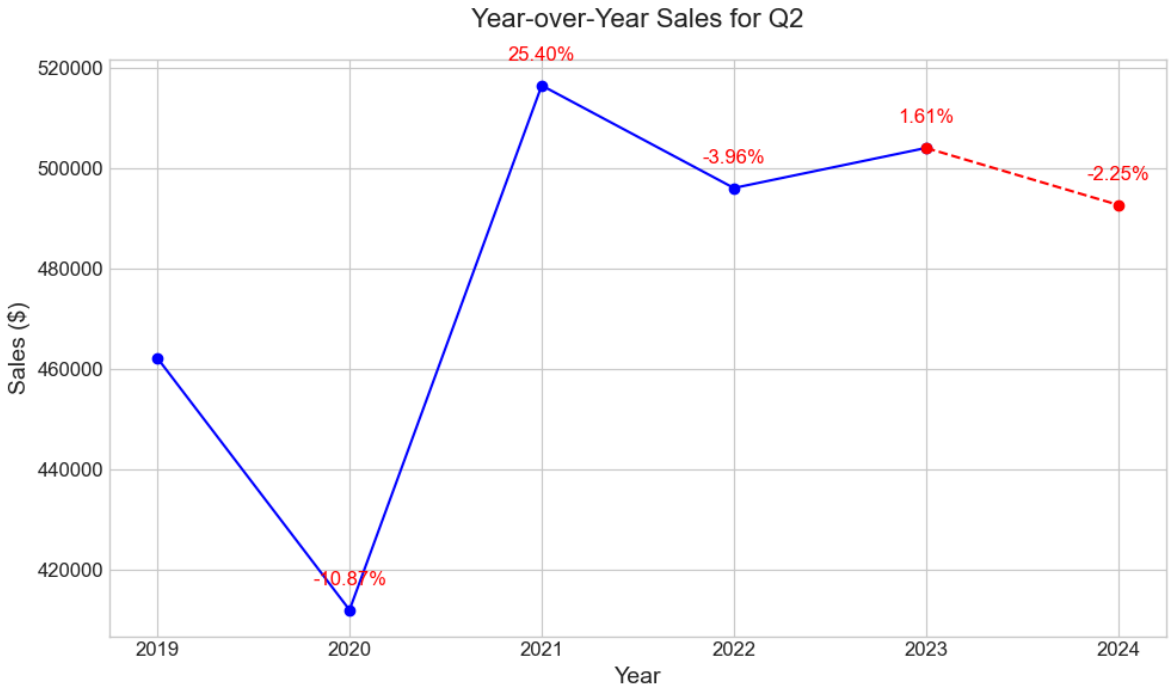
Output:

```
Enter the number of previous years you want to analyze: 5
old   1: SELECT &PreviousYear - &NumYears + 1 AS StartYear FROM dual
new   1: SELECT       2023 - 5 + 1 AS StartYear FROM dual
old   9: WHERE D.cal_year BETWEEN &StartYear AND &PreviousYear
new   9: WHERE D.cal_year BETWEEN       2019 AND       2023
View created.
old  12:     WHERE Q.cal_year BETWEEN &StartYear AND &PreviousYear
new  12:     WHERE Q.cal_year BETWEEN       2019 AND       2023
                    ====================================================================
                    Year-over-Year Quarterly Sales Growth Analysis And Projection (2024)
                    ====================================================================
DATE: 20-SEP-24
PAGE:    1
                                                    Absolute              Projected     Projected
QUARTER  YEAR1    SALESYEAR1    SALESYEAR2          Growth Growth %    Sales 2024 Growth 2024 %
-------- ----- ------------- ------------- ------------- -------- ------------- -------------
Q1        2019   $481,125.91                                        $478,575.60
          2020   $525,470.04   $481,125.91    $44,344.13     9.22
          2021   $456,662.39   $525,470.04   -$68,807.65   -13.09
          2022   $471,630.94   $456,662.39    $14,968.55     3.28
          2023   $467,214.70   $471,630.94    -$4,416.24     -.94                          2.43
********       -------------                -------------- -------- *************
Average:         $480,420.80                  -$3,477.80     -.38

Q2        2019   $462,105.83                                        $492,650.82
          2020   $411,868.87   $462,105.83   -$50,236.96   -10.87
          2021   $516,469.13   $411,868.87   $104,600.26    25.40
          2022   $496,021.35   $516,469.13   -$20,447.78    -3.96
          2023   $504,014.19   $496,021.35     $7,992.84     1.61                         -2.25
********       -------------                -------------- -------- *************
Average:         $478,095.87                  $10,477.09     3.04
```

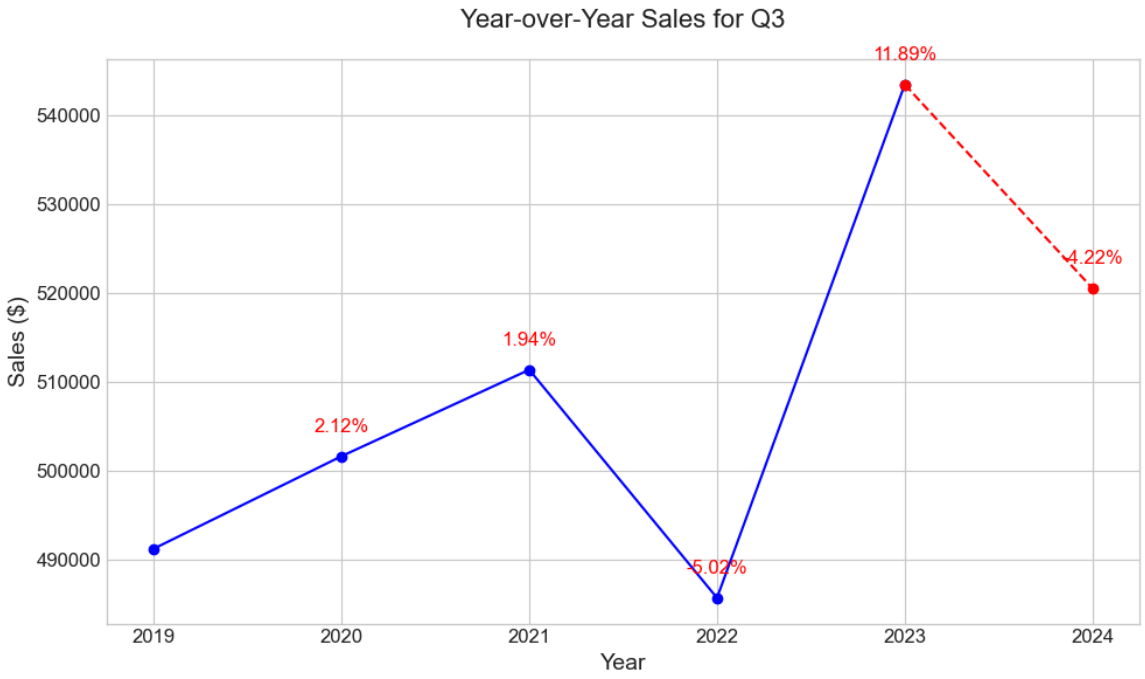
```
Q3        2019   $491,247.24                                  $520,537.58
          2020   $501,653.46   $491,247.24    $10,406.22     2.12
          2021   $511,382.10   $501,653.46     $9,728.64     1.94
          2022   $485,719.69   $511,382.10   -$25,662.41    -5.02
          2023   $543,467.59   $485,719.69    $57,747.90    11.89                      -4.22
********       -------------              ------------- -------- *************
Average:         $506,694.02                $13,055.09     2.73

Q4        2019   $507,082.22                                  $477,975.71
          2020   $492,322.57   $507,082.22   -$14,759.65    -2.91
          2021   $462,463.08   $492,322.57   -$29,859.49    -6.07
          2022   $463,915.81   $462,463.08     $1,452.73      .31
          2023   $486,706.04   $463,915.81    $22,790.23     4.91                      -1.79
********       -------------              ------------- -------- *************
Average:         $482,497.94                -$5,094.05     -.94

20 rows selected.
```



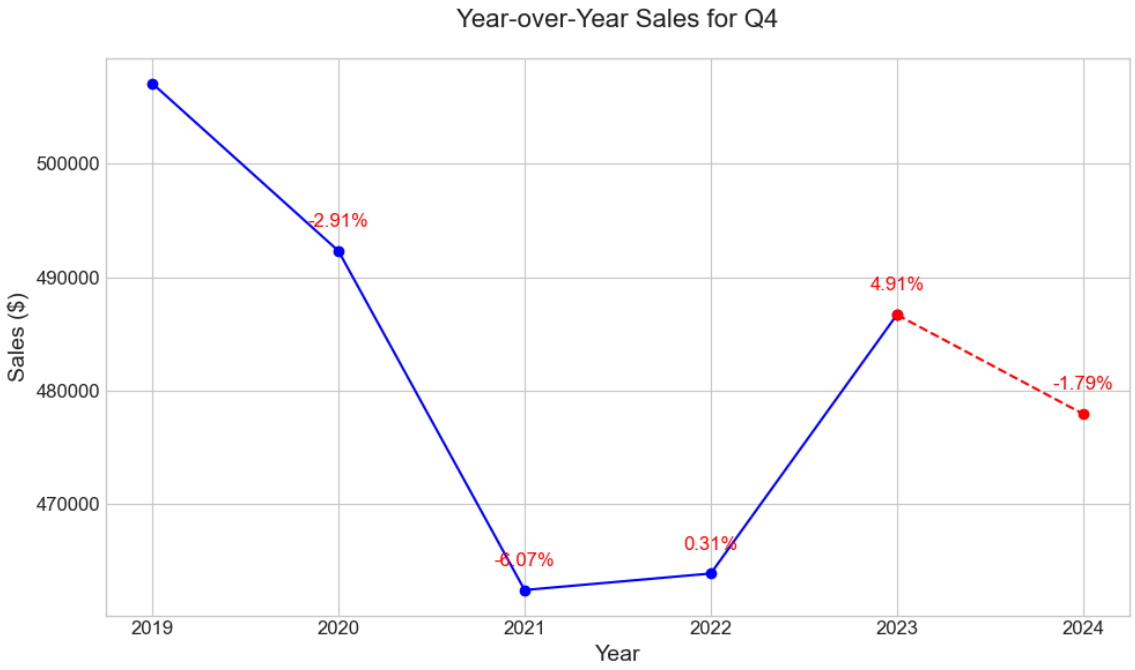
Linechart 1: Sales amount of Quarter 1 from 2019 to 2023 and projected sales amount of 2024



Linechart 2: Sales amount of Quarter 2 from 2019 to 2023 and projected sales amount of 2024



Linechart 3: Sales amount of Quarter 3 from 2019 to 2023 and projected sales amount of 2024



Linechart 4: Sales amount of Quarter 4 from 2019 to 2023 and projected sales amount of 2024

This query projects the sales for each quarter in 2024 to provide insights based on historical sales data and average growth rates, enabling businesses to forecast revenue and set realistic targets. For Q1 2024, sales are expected to grow by 2.43% compared to Q1 2023, establishing a solid foundation for further growth. Businesses should capitalize on this by strengthening early-year marketing efforts, boosting customer engagement, and leveraging seasonal promotions or product launches. However, projections for Q2, Q3, and Q4 indicate declines of 2.25%, 4.22%, and 1.79%, respectively. To address these declines, businesses should optimize pricing, adjust inventory based on demand, and focus on mid-year and holiday promotions. Introducing new products, offering exclusive discounts, and enhancing customer loyalty programs will be essential to recover sales in the later quarters. The projected sales amounts for each quarter are calculated by applying the average growth percentage to the average sales from previous years as follows: Q1 average sales = $480,420.80 (growth = -0.38%), Q2 = $478,095.87 (growth = 3.04%), Q3 = $506,694.02 (growth = 2.73%), and Q4 = $482,497.94 (growth = -0.94%). Projected sales for each quarter are determined using the **formula: average sales * (1 + average growth/100).**

3.2.2 Threshold-Based Analysis of Weekend vs. Weekday Sales Performance During a Promotional Period by Menu Category

SQL Code:

```
SET LINESIZE 80
SET PAGESIZE 50

-- Column Settings
COLUMN Avg_Weekday_Sales FORMAT $9,999.99 HEADING
'AVG|WEEKDAY|SALES'
COLUMN AVG_WEEKEND_SALES FORMAT $9,999.99 HEADING
'AVG|WEEKEND|SALES'
COLUMN "WEEKEND-WEEKDAY_SALES" FORMAT $9,999.99 HEADING
'WEEKEND|-WEEKDAY|SALES|DIFF'
COLUMN "WEEKEND-WEEKDAY_SALES(%)" FORMAT 999.99 HEADING
'WEEKEND|-WEEKDAY|SALES|DIFF(%)'
COLUMN CATEGORYNAME FORMAT A12
COLUMN TOTAL_SALES FORMAT $999,999.99
COLUMN DAY_TYPE FORMAT A8
COLUMN "Required_Sales_For_Uplift" FORMAT A25

-- Prompt for promo ID and uplift percentage
ACCEPT promo_id PROMPT 'Enter Promo ID: '
ACCEPT uplift_percentage PROMPT 'Enter desired uplift percentage
(e.g., 10 for 10%): '

-- TOTAL REVENUE SALES
-- DATA SET 1
CREATE OR REPLACE VIEW Total_Revenue_Sales AS
SELECT M.categoryName,
       SUM(QUANTITY) AS QTY_SOLD,
       SUM(linetotal) AS Total_Revenue
FROM salesfact SF
JOIN PROMOTION_DIM P ON SF.PROMO_KEY = P.PROMO_KEY
JOIN date_dim D ON SF.date_key = D.date_key
JOIN MENU_dim M ON SF.MENU_key = M.MENU_key
WHERE P.PROMOID = &promo_id
AND D.CAL_DATE BETWEEN P.PROMOSTARTDATE AND P.PROMOENDDATE
GROUP BY M.categoryName
ORDER BY M.categoryName, Total_Revenue DESC;

-- WEEKDAY
CREATE OR REPLACE VIEW WEEK_DAY_Sales AS
SELECT M.categoryname, SUM(QUANTITY) AS WEEK_DAY_QTY,
SUM(LINETOTAL) AS WEEK_DAY_SALES
FROM SALESFACT SF
```

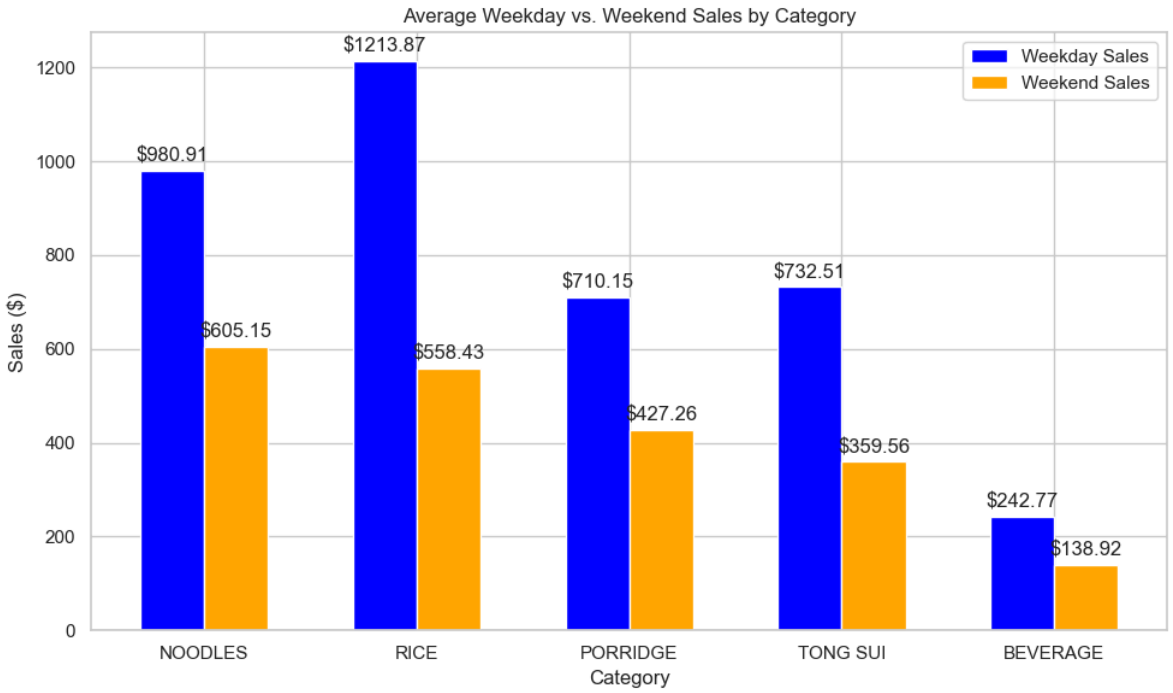
```
JOIN PROMOTION_DIM P ON SF.PROMO_KEY = P.PROMO_KEY
JOIN MENU_DIM M ON SF.MENU_KEY = M.MENU_KEY
JOIN DATE_DIM D ON SF.DATE_KEY = D.DATE_KEY
WHERE P.PROMOID = &promo_id
AND D.CAL_DATE BETWEEN P.PROMOSTARTDATE AND P.PROMOENDDATE
AND WEEKDAY_IND = 'Y'
GROUP BY M.categoryname
ORDER BY WEEK_DAY_SALES DESC;

-- WEEKEND
CREATE OR REPLACE VIEW WEEK_END_Sales AS
SELECT M.categoryname, SUM(QUANTITY) AS WEEK_END_QTY,
SUM(LINETOTAL) AS WEEK_END_SALES
FROM SALESFACT SF
JOIN PROMOTION_DIM P ON SF.PROMO_KEY = P.PROMO_KEY
JOIN MENU_DIM M ON SF.MENU_KEY = M.MENU_KEY
JOIN DATE_DIM D ON SF.DATE_KEY = D.DATE_KEY
WHERE P.PROMOID = &promo_id
AND D.CAL_DATE BETWEEN P.PROMOSTARTDATE AND P.PROMOENDDATE
AND WEEKDAY_IND = 'N'
GROUP BY M.categoryname
ORDER BY WEEK_END_SALES DESC;

-- COMBINE
CREATE OR REPLACE VIEW Combined_Sales AS
SELECT TRS.*,
       WDS.week_day_SALES,
       WDS.WEEK_DAY_QTY,
       WES.week_End_SALES,
       WES.WEEK_END_QTY
FROM Total_Revenue_Sales TRS
JOIN WEEK_DAY_Sales WDS ON TRS.categoryName = WDS.categoryName
JOIN WEEK_END_Sales WES ON TRS.categoryName = WES.categoryName;

-- PROMO DATES
CREATE OR REPLACE VIEW PROMODATES AS
SELECT
        P.PROMOSTARTDATE,
        P.PROMOENDDATE
FROM PROMOTION_DIM P
WHERE P.PROMOID = &promo_id;

-- COUNT WEEKDAYS, WEEKENDS
CREATE OR REPLACE VIEW NO_OF_DAYS AS
SELECT
```

```
        COUNT(CASE WHEN D.WEEKDAY_IND = 'Y' THEN 1 END) AS
Num_Weekdays,
        COUNT(CASE WHEN D.WEEKDAY_IND = 'N' THEN 1 END) AS
Num_Weekends
FROM DATE_DIM D
JOIN PromoDates PD ON D.CAL_DATE BETWEEN PD.PROMOSTARTDATE AND
PD.PROMOENDDATE;

-- REPORT
TTITLE ON
TTITLE CENTER
'================================================================
==' SKIP 1 -
       CENTER ' Weekend vs. Weekday Sales Performance During a
Promotional Period ' SKIP 1 -
       CENTER
'================================================================
==' SKIP 1 LEFT 'DATE: ' _DATE SKIP 1 LEFT  'PAGE: ' FORMAT 999
SQL.PNO SKIP 1


SELECT
    TRS.categoryName,
    WDS.week_day_SALES / NULLIF(NOD.Num_Weekdays, 0) AS
Avg_Weekday_Sales,
    WES.week_End_SALES / NULLIF(NOD.Num_Weekends, 0) AS
Avg_Weekend_Sales,
    (WES.week_End_SALES / NULLIF(NOD.Num_Weekends, 0)) -
(WDS.week_day_SALES / NULLIF(NOD.Num_Weekdays, 0)) AS
"Weekend-Weekday_Sales",
    CASE
        WHEN WDS.week_day_SALES / NULLIF(NOD.Num_Weekdays, 0) =
0 THEN 0
        ELSE ROUND((((WES.week_End_SALES /
NULLIF(NOD.Num_Weekends, 0)) - (WDS.week_day_SALES /
NULLIF(NOD.Num_Weekdays, 0))) /
              (WDS.week_day_SALES / NULLIF(NOD.Num_Weekdays,
0))) * 100,2)
    END AS "Weekend-Weekday_Sales(%)",
    CASE
        WHEN ABS(ROUND((((WES.week_End_SALES /
NULLIF(NOD.Num_Weekends, 0)) - (WDS.week_day_SALES /
NULLIF(NOD.Num_Weekdays, 0))) /
              (WDS.week_day_SALES / NULLIF(NOD.Num_Weekdays,
0))) * 100,2)) >= 40
```

```
        THEN
            CASE
                WHEN (WDS.week_day_SALES /
NULLIF(NOD.Num_Weekdays, 0)) < (WES.week_End_SALES /
NULLIF(NOD.Num_Weekends, 0)) THEN
                    'Weekday:' ||
TO_CHAR(ROUND((WDS.week_day_SALES / NULLIF(NOD.Num_Weekdays, 0))
* (1 + &uplift_percentage / 100), 2), '$9999.99')
                ELSE
                    'Weekend:' ||
TO_CHAR(ROUND((WES.week_End_SALES / NULLIF(NOD.Num_Weekends, 0))
* (1 + &uplift_percentage / 100), 2), '$9999.99')
            END
        ELSE 'N/A'
    END AS "Required_Sales_For_Uplift"
FROM
    Total_Revenue_Sales TRS
JOIN
    WEEK_DAY_Sales WDS ON TRS.categoryName = WDS.categoryName
JOIN
    WEEK_END_Sales WES ON TRS.categoryName = WES.categoryName
JOIN
    NO_OF_DAYS NOD ON 1 = 1;

CLEAR COLUMNS
TTITLE OFF
```

Output:

```
SQL> @"D:\RDSY2S3\DataWareHouse\DW Asgm\Queries\SETTLE\try_query2_trytry.txt"
Enter Promo ID: 79
Enter desired uplift percentage (e.g., 10 for 10%): 20

View created.


View created.


View created.


View created.


View created.


View created.

        ================================================================
        Weekend vs. Weekday Sales Performance During a Promotional Period
        ================================================================
DATE: 21-SEP-24
PAGE:    1
                                      WEEKEND  WEEKEND
                    AVG        AVG   -WEEKDAY -WEEKDAY
                WEEKDAY    WEEKEND      SALES    SALES
CATEGORYNAME      SALES      SALES       DIFF  DIFF(%) Required_Sales_For_Uplift
------------ ---------- ---------- ---------- -------- -------------------------
NOODLES         $980.91    $605.15   -$375.76   -38.31 N/A
RICE          $1,213.87    $558.43   -$655.45   -54.00 Weekend:  $670.11
PORRIDGE        $710.15    $427.26   -$282.89   -39.83 N/A
TONG SUI        $732.51    $359.56   -$372.95   -50.91 Weekend:  $431.47
BEVERAGE        $242.77    $138.92   -$103.85   -42.78 Weekend:  $166.70
```



This query aims to compare sales performance between weekdays and weekends during a promotional period across different menu categories. By calculating the average sales for both weekdays and weekends, it highlights the percentage differences in sales between these periods. The goal is to identify significant variations, particularly when weekend sales underperform relative to weekdays. If the sales difference exceeds a 40% threshold, the query suggests a "Required Sales for Uplift" to meet a user-defined percentage uplift target. For instance, in the RICE category, where weekend sales are 54% lower than weekday sales, the query calculates that weekend sales need to reach $670.11 to achieve the desired uplift. Similarly, for the BEVERAGE category, where weekend sales are 42.78% lower, the uplift target is $166.70 for the weekend. This analysis helps decision-makers determine actionable sales goals and identifies opportunities for differentiated promotions, such as weekend or weekday-exclusive promotions, tailored to each category's performance. Hence, it is concluded that weekend-exclusive promotions for RICE, TONG SUI, and BEVERAGE are required to boost weekend sales, with the desired uplift sales target to be achieved during the next similar promotion.

3.2.3 New Year vs Non-New Year Sales Analysis over past n years

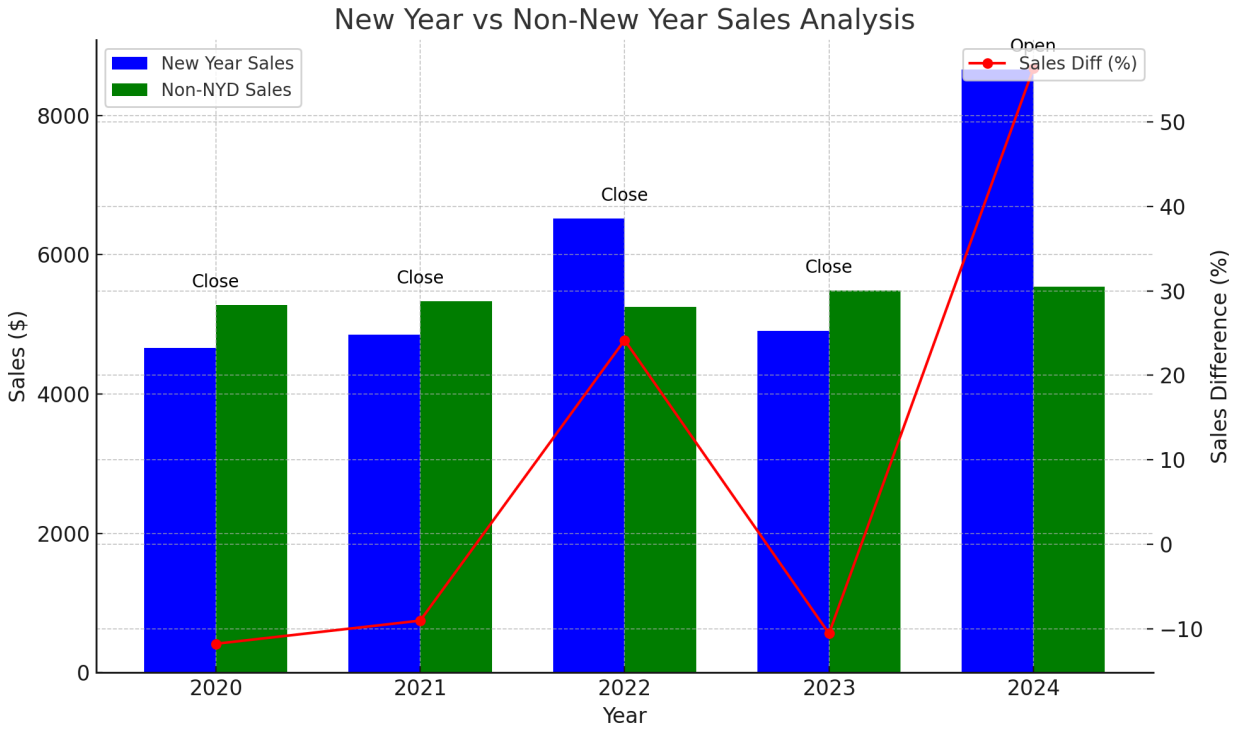
SQL Code:

```
SET PAGESIZE 50
SET LINESIZE 70
SET VERIFY OFF
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';

COLUMN cal_year FORMAT 9999 HEADING 'YEAR'
COLUMN avg_non_nyd_sales FORMAT $999,999.99 HEADING
'AVG|NON-NYD|SALES'
COLUMN nyd_sales FORMAT $999,999.99 HEADING 'NEW YEAR|SALES'
COLUMN SALES_DIFFERENCE FORMAT $999,999.99  HEADING
'SALES|DIFFERENCE'
COLUMN perct_of_nyd_to_non_nyd FORMAT 999.99 HEADING '(%)|NYD
to|Non-NYD'

-- Prompt for input to specify how many years of data to analyze
ACCEPT num_years NUMBER PROMPT 'Enter the number of years to
analyze: '

TTITLE ON
TTITLE CENTER '=========================================' SKIP
1 -
       CENTER '  NEW YEAR VS NON-NEW YEAR SALES ANALYSIS ' SKIP
1 -
       CENTER '  TO DETERMINE OPEN/CLOSE NEXT YEAR' SKIP 1 -
       CENTER '=========================================' SKIP
1 LEFT 'DATE: ' _DATE SKIP 1 LEFT  'PAGE: ' FORMAT 999 SQL.PNO
SKIP 1

WITH SALES_DATASET AS (
    SELECT
        D.CAL_YEAR,
        CASE
            WHEN TO_CHAR(D.CAL_DATE, 'MM-DD') = '01-01' THEN
'New Year'
            ELSE 'Non-New Year'
        END AS SEASON_TYPE,
        COUNT(DISTINCT d.date_key) AS NUM_OF_DAYS,
        SUM(SF.LINETOTAL) AS total_sales
    FROM SALESFACT SF
    JOIN DATE_DIM D
        ON SF.date_key = D.date_key
```

```
    WHERE D.CAL_YEAR BETWEEN (EXTRACT(YEAR FROM TRUNC(SYSDATE))
- &num_years + 1) AND EXTRACT(YEAR FROM TRUNC(SYSDATE))
    GROUP BY D.CAL_YEAR,
             CASE
                 WHEN TO_CHAR(D.CAL_DATE, 'MM-DD') = '01-01'
THEN 'New Year'
                 ELSE 'Non-New Year'
             END
),
COMPARE AS (
    SELECT
        CAL_YEAR,
        MAX(CASE WHEN SEASON_TYPE = 'New Year' THEN total_sales
ELSE NULL END) AS NYD_SALES,
        AVG(CASE WHEN SEASON_TYPE = 'Non-New Year' THEN
total_sales / NUM_OF_DAYS ELSE NULL END) AS AVG_NON_NYD_SALES,
        MAX(CASE WHEN SEASON_TYPE = 'Non-New Year' THEN
NUM_OF_DAYS ELSE NULL END) AS NON_NYD_DAYS
    FROM SALES_DATASET
    GROUP BY CAL_YEAR
)
SELECT
    CAL_YEAR,
    NYD_SALES,
    AVG_NON_NYD_SALES,
    (NYD_SALES - AVG_NON_NYD_SALES) AS SALES_DIFFERENCE,
    CASE
        WHEN AVG_NON_NYD_SALES = 0 THEN 0
        ELSE ROUND(
            (((NYD_SALES - AVG_NON_NYD_SALES)/
AVG_NON_NYD_SALES) * 100), 2
        )
    END AS perct_of_nyd_to_non_nyd,
    CASE
        WHEN (((NYD_SALES - AVG_NON_NYD_SALES)/
AVG_NON_NYD_SALES) * 100) > 50 THEN 'Consider Open'
        ELSE 'Consider Close'
    END AS SUGGESTION
FROM COMPARE
ORDER BY CAL_YEAR;

CLEAR COLUMNS
TTITLE OFF;
```

Output:

```
SQL> @D:\RDSY2S3\RDS2S3G3_WYE_YJ_ARJKA_TWY_WZN\WongYeeEn\WYE_query3.txt

Session altered.

Enter the number of years to analyze: 5

                ==========================================
                  NEW YEAR VS NON-NEW YEAR SALES ANALYSIS
                      TO DETERMINE OPEN/CLOSE NEXT YEAR
                ==========================================
DATE: 21/09/2024
PAGE:    1
                                AVG                    (%)
           NEW YEAR       NON-NYD         SALES   NYD to
 YEAR         SALES         SALES    DIFFERENCE Non-NYD SUGGESTION
----- ------------ ------------ ------------ ------- --------------
 2020    $4,658.10    $5,278.51     -$620.41  -11.75 Consider Close
 2021    $4,854.79    $5,335.50     -$480.71   -9.01 Consider Close
 2022    $6,518.00    $5,249.37    $1,268.63   24.17 Consider Close
 2023    $4,906.80    $5,484.88     -$578.08  -10.54 Consider Close
 2024    $8,656.00    $5,537.83    $3,118.17   56.31 Consider Open
```



This query compares New Year sales with average non-New Year sales for past n years where n is the number of years that the user wants to analyze to determine whether it is profitable to stay open for the coming New Year.

In 2020, the hawker center generated $4,658.10 in New Year sales, which was 11.75% lower than the average non-New Year sales ($5,278.51). Since this figure falls short of the 50% threshold, the suggestion is to ‘Consider Close’ due to the weaker sales performance during the New Year period.

In 2021, the hawker center generated $4,854.79 in New Year sales, which is 9.01% less than the average non-New Year sales ($5,335.50). Since the 50% threshold was not reached, the suggestion is to ‘Consider Close.’ In 2022, New Year sales were $6,518.00, exceeding non-New Year sales by 24.17%, but still not reaching the threshold. The suggestion is to ‘Consider Close.’

In 2023, New Year sales were $4,906.80, which was 10.54% lower than the average non-New Year sales ($5,484.88). The suggestion remains ‘Consider Close.’ In 2024, the hawker center saw a significant improvement with $8,656.00 in New Year sales, 56.31% higher than non-New Year sales, meeting the threshold. The suggestion is to ‘Consider Open.’

In conclusion, since the suggestion is to "Consider Close" in four out of five years, the hawker center should consider closing for the coming New Year.

3.3 Ashantha Rosary James K Arokiasamy

3.3.1 Evaluate total quantity sold in year 2023 by quarter for inventory stock up

SQL Code:

```
SET PAGESIZE 39
SET LINESIZE 120
SET VERIFY OFF

TTITLE ON
TTITLE LEFT '
+-----------------------------------------------------------------
----------------------+' SKIP 1 -
       LEFT '        |                      TOTAL QUANTITY SOLD
IN YEAR 2023 BY QUARTER                        |' SKIP 1 -
       LEFT '
+-----------------------------------------------------------------
----------------------+' SKIP 2 -
       LEFT 'DATE: ' _DATE SKIP 1 -
       LEFT 'PAGE: ' FORMAT 999 SQL.PNO SKIP 2 -

BREAK ON CATEGORY SKIP 1
COLUMN CATEGORY HEADING "CATEGORY" FORMAT A10
COLUMN "MENU_ITEM" HEADING "MENU NAME" FORMAT A27
COLUMN TOTAL_QTY_SOLD_2023 HEADING "TOTAL|SOLD |2023 " FORMAT
99999
COLUMN Q1 HEADING "Q1" FORMAT 9999
COLUMN Q2 HEADING "Q2" FORMAT 9999
COLUMN Q3 HEADING "Q3" FORMAT 9999
COLUMN Q4 HEADING "Q4" FORMAT 9999
COLUMN AVG_SOLD_QUARTER HEADING "AVG|SOLD|QUARTER" FORMAT 99999
COLUMN TOTAL_QTY_SOLD_2022 HEADING "TOTAL|SOLD |2022 " FORMAT
999999
COLUMN PERCENTAGE_CHANGE HEADING "% CHANGE" FORMAT 999.99
COLUMN STOCKUP HEADING "STOCKUP" FORMAT 99999

-- View with detailed sales data for 2023 by quarter
CREATE OR REPLACE VIEW MenuSales2023 AS
SELECT MD.CATEGORYNAME,MD.MENUNAME AS MENU_ITEM,
       SUM(SF.Quantity) AS TOTAL_QTY_SOLD,
       SUM(CASE WHEN DD.CAL_QUARTER = 'Q1' THEN SF.Quantity ELSE
0 END) AS Q1,
       SUM(CASE WHEN DD.CAL_QUARTER = 'Q2' THEN SF.Quantity ELSE
0 END) AS Q2,
       SUM(CASE WHEN DD.CAL_QUARTER = 'Q3' THEN SF.Quantity ELSE
0 END) AS Q3,
```

```
       SUM(CASE WHEN DD.CAL_QUARTER = 'Q4' THEN SF.Quantity ELSE
0 END) AS Q4,
       SUM(SF.Quantity) / 4 AS AVG_SOLD_QUARTER
FROM SALESFACT SF
JOIN MENU_DIM MD ON SF.MENU_KEY = MD.MENU_KEY
JOIN DATE_DIM DD ON SF.DATE_KEY = DD.DATE_KEY
WHERE DD.CAL_YEAR = 2023
GROUP BY MD.CATEGORYNAME, MD.MENUNAME;

-- View with sales data for 2022
CREATE OR REPLACE VIEW MenuSales2022 AS
SELECT MD.MENUNAME AS MENU_ITEM,
       SUM(SF.Quantity) AS TOTAL_QTY_SOLD_2022
FROM SALESFACT SF
JOIN MENU_DIM MD ON SF.MENU_KEY = MD.MENU_KEY
JOIN DATE_DIM DD ON SF.DATE_KEY = DD.DATE_KEY
WHERE DD.CAL_YEAR = 2022
GROUP BY MD.MENUNAME;

-- Combined view with sales comparison and adjustments
CREATE OR REPLACE VIEW MenuSalesComparison AS
SELECT M23.CATEGORYNAME AS CATEGORY,
       M23.MENU_ITEM,
       M23.Q1, M23.Q2, M23.Q3, M23.Q4,
       M23.TOTAL_QTY_SOLD AS TOTAL_QTY_SOLD_2023,
       M23.AVG_SOLD_QUARTER,
       M22.TOTAL_QTY_SOLD_2022, -- Removed COALESCE
       CASE WHEN M22.TOTAL_QTY_SOLD_2022 IS NULL THEN NULL
            ELSE (M23.TOTAL_QTY_SOLD - M22.TOTAL_QTY_SOLD_2022)
* 100.0 / M22.TOTAL_QTY_SOLD_2022 END AS PERCENTAGE_CHANGE,
       CASE WHEN M23.TOTAL_QTY_SOLD > M22.TOTAL_QTY_SOLD_2022
THEN M23.AVG_SOLD_QUARTER * 1.2
            ELSE M23.AVG_SOLD_QUARTER END AS STOCKUP
FROM MenuSales2023 M23
JOIN MenuSales2022 M22 ON M23.MENU_ITEM = M22.MENU_ITEM
ORDER BY M23.CATEGORYNAME, M23.TOTAL_QTY_SOLD DESC;

SELECT * FROM MenuSalesComparison;

CLEAR COLUMNS
CLEAR BREAK
TTITLE OFF;
```

Output :

TOTAL QUANTITY SOLD IN YEAR 2023 BY QUARTER

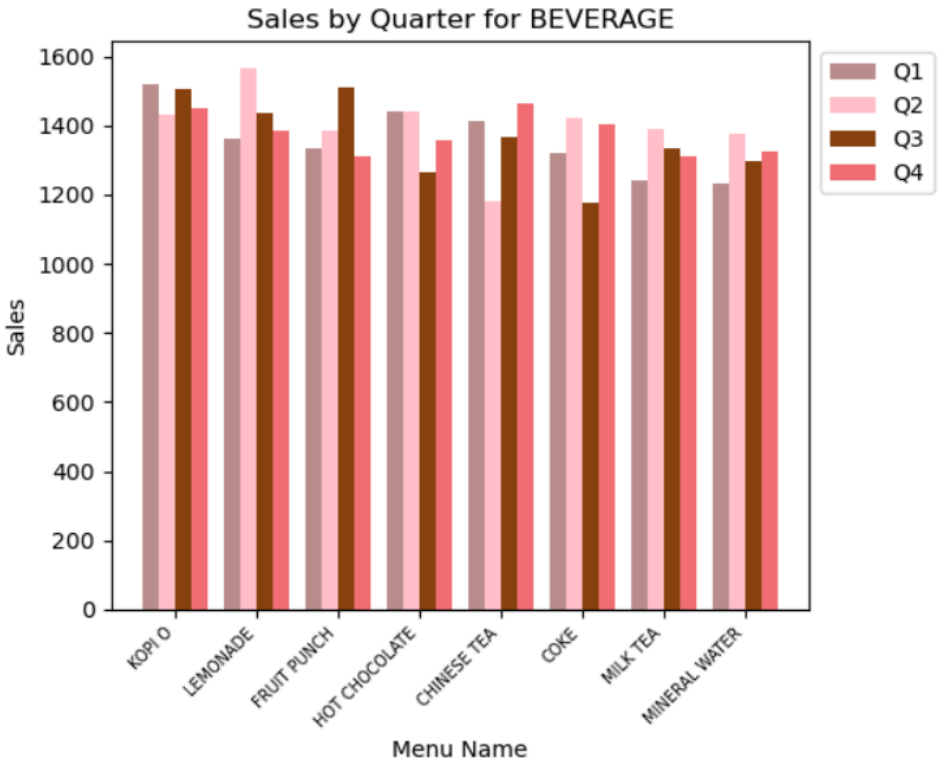
DATE: 19-SEP-24
PAGE: 1

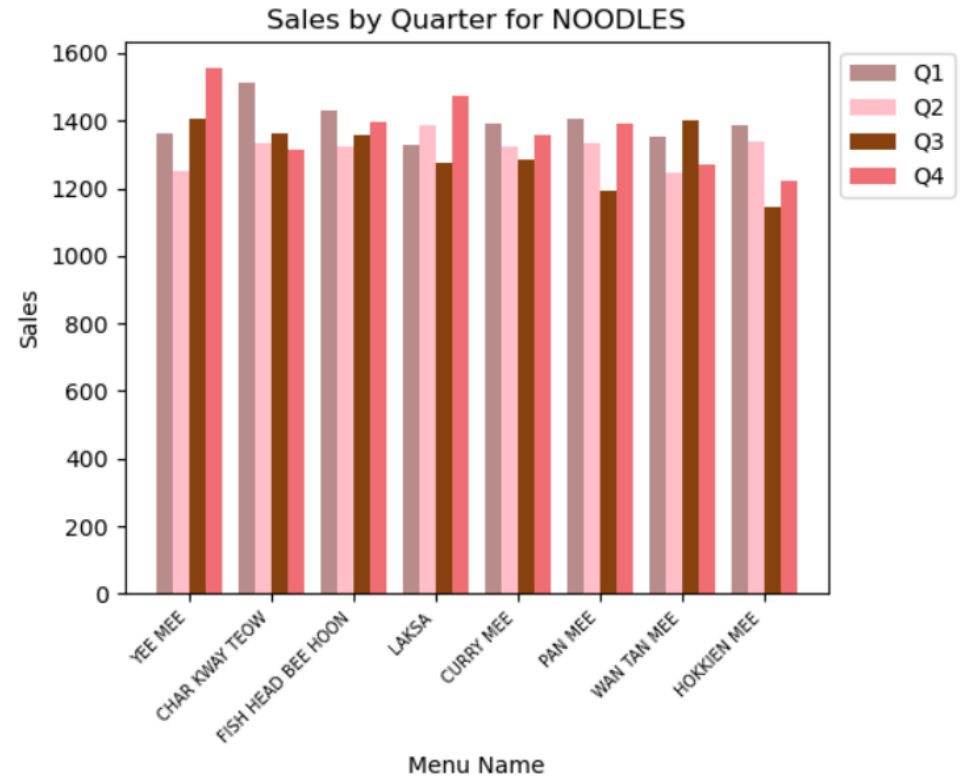
CATEGORY	MENU NAME	Q1	Q2	Q3	Q4	TOTAL SOLD 2023	AVG SOLD QUARTER	TOTAL SOLD 2022	% CHANGE	STOCKUP
BEVERAGE	KOPI O	1521	1431	1506	1450	5908	1477	5183	13.99	1772
	LEMONADE	1363	1565	1434	1385	5747	1437	5577	3.05	1724
	FRUIT PUNCH	1333	1387	1509	1309	5538	1385	5487	.93	1661
	HOT CHOCOLATE	1439	1439	1265	1359	5502	1376	5381	2.25	1651
	CHINESE TEA	1414	1181	1365	1466	5426	1357	5145	5.46	1628
	COKE	1318	1420	1176	1403	5317	1329	5436	-2.19	1329
	MILK TEA	1242	1390	1335	1312	5279	1320	5709	-7.53	1320
	MINERAL WATER	1231	1376	1295	1324	5226	1307	5406	-3.33	1307
NOODLES	YEE MEE	1362	1251	1407	1554	5574	1394	5476	1.79	1672
	CHAR KWAY TEOW	1511	1331	1360	1315	5517	1379	5606	-1.59	1379
	FISH HEAD BEE HOON	1430	1322	1356	1397	5505	1376	5343	3.03	1652
	LAKSA	1326	1384	1276	1473	5459	1365	5434	.46	1638
	CURRY MEE	1390	1323	1283	1355	5351	1338	5218	2.55	1605
	PAN MEE	1406	1334	1195	1390	5325	1331	5270	1.04	1598
	WAN TAN MEE	1352	1245	1402	1271	5270	1318	5328	-1.09	1318
	HOKKIEN MEE	1386	1340	1143	1220	5089	1272	5599	-9.11	1272
PORRIDGE	CENTURY EGG PORRIDGE	1373	1372	1399	1388	5532	1383	5612	-1.43	1383
	LOTUS ROOT PORRIDGE	1388	1410	1455	1270	5523	1381	5246	5.28	1657
	SWEET CORN PORRIDGE	1327	1456	1353	1366	5502	1376	5506	-.07	1376
	GINGER PORRIDGE	1381	1308	1282	1476	5447	1362	5935	-8.22	1362
	SWEET POTATO PORRIDGE	1302	1397	1386	1280	5365	1341	5307	1.09	1610
	PORK BONE PORRIDGE	1390	1211	1331	1424	5356	1339	5418	-1.14	1339
	CHICKEN PORRIDGE	1290	1206	1369	1377	5242	1311	5440	-3.64	1311
	MUSHROOM PORRIDGE	1410	1275	1287	1239	5211	1303	5401	-3.52	1303

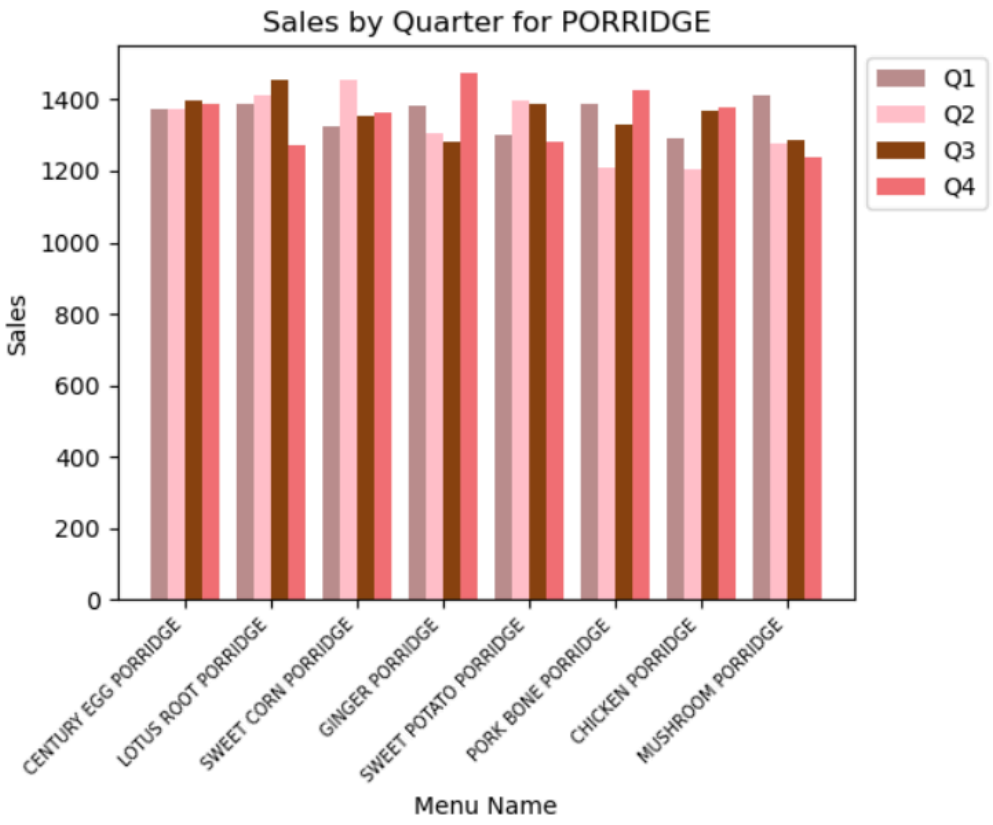
TOTAL QUANTITY SOLD IN YEAR 2023 BY QUARTER

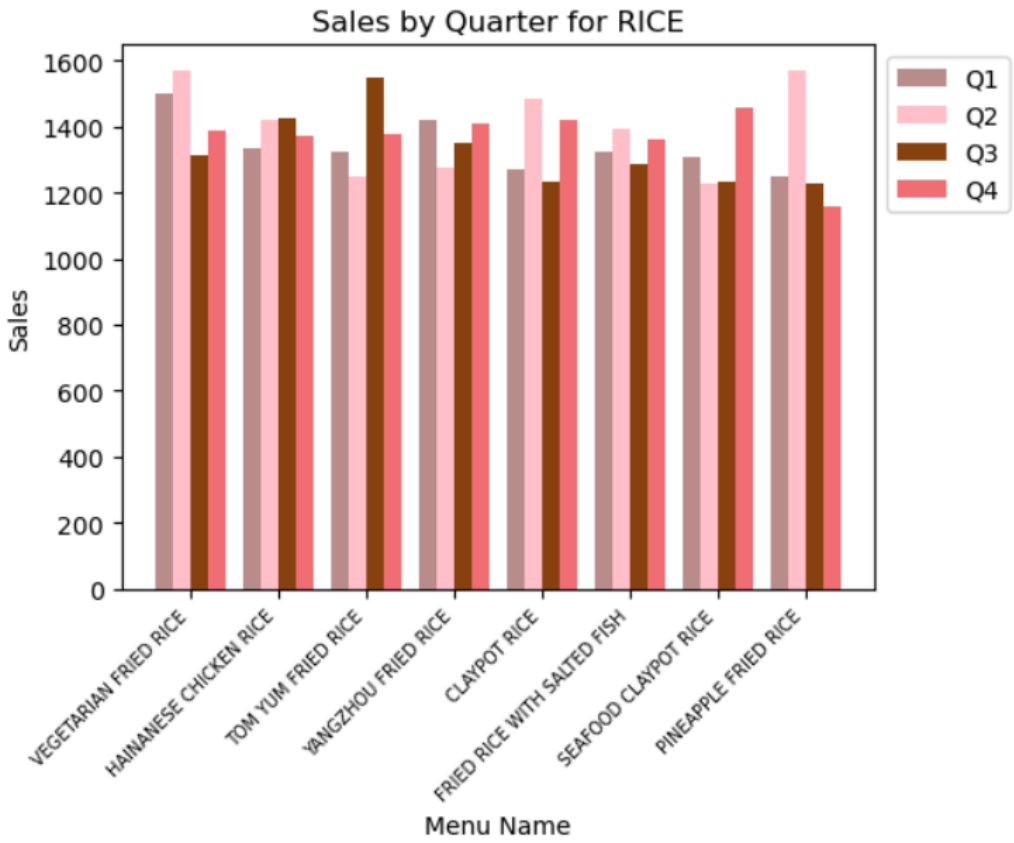
DATE: 19-SEP-24
PAGE: 2

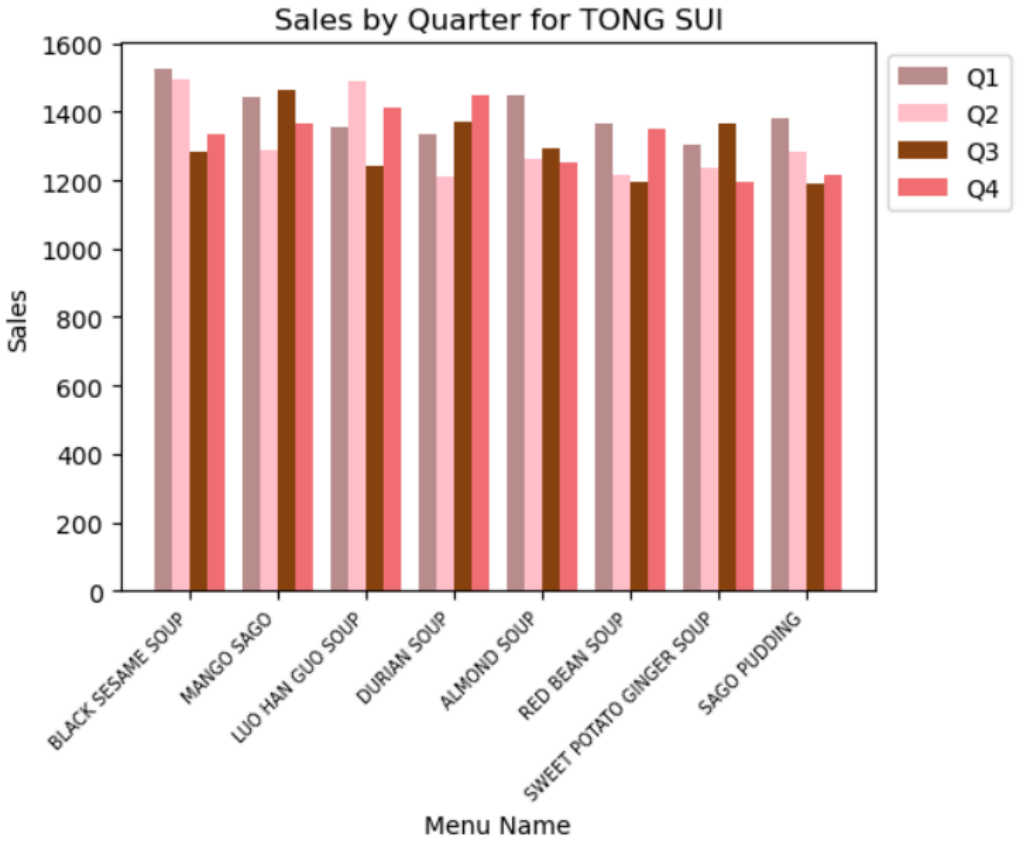
CATEGORY	MENU NAME	Q1	Q2	Q3	Q4	TOTAL SOLD 2023	AVG SOLD QUARTER	TOTAL SOLD 2022	% CHANGE	STOCKUP
RICE	VEGETARIAN FRIED RICE	1501	1569	1311	1385	5766	1442	5397	6.84	1730
	HAINANESE CHICKEN RICE	1332	1419	1427	1371	5549	1387	6026	-7.92	1387
	TOM YUM FRIED RICE	1324	1251	1548	1376	5499	1375	5720	-3.86	1375
	YANGZHOU FRIED RICE	1418	1276	1348	1409	5451	1363	5172	5.39	1635
	CLAYPOT RICE	1269	1484	1234	1419	5406	1352	5105	5.90	1622
	FRIED RICE WITH SALTED FISH	1321	1395	1284	1363	5363	1341	5165	3.83	1609
	SEAFOOD CLAYPOT RICE	1305	1226	1231	1455	5217	1304	5226	-.17	1304
	PINEAPPLE FRIED RICE	1247	1568	1228	1159	5202	1301	5634	-7.67	1301
TONG SUI	BLACK SESAME SOUP	1525	1496	1281	1332	5634	1409	5306	6.18	1690
	MANGO SAGO	1443	1289	1462	1364	5558	1390	5475	1.52	1667
	LUO HAN GUO SOUP	1356	1489	1240	1413	5498	1375	5534	-.65	1375
	DURIAN SOUP	1336	1211	1369	1450	5366	1342	5493	-2.31	1342
	ALMOND SOUP	1446	1260	1291	1250	5247	1312	5315	-1.28	1312
	RED BEAN SOUP	1363	1216	1194	1349	5122	1281	5592	-8.40	1281
	SWEET POTATO GINGER SOUP	1301	1238	1364	1195	5098	1275	5606	-9.06	1275
	SAGO PUDDING	1380	1285	1188	1218	5071	1268	5352	-5.25	1268

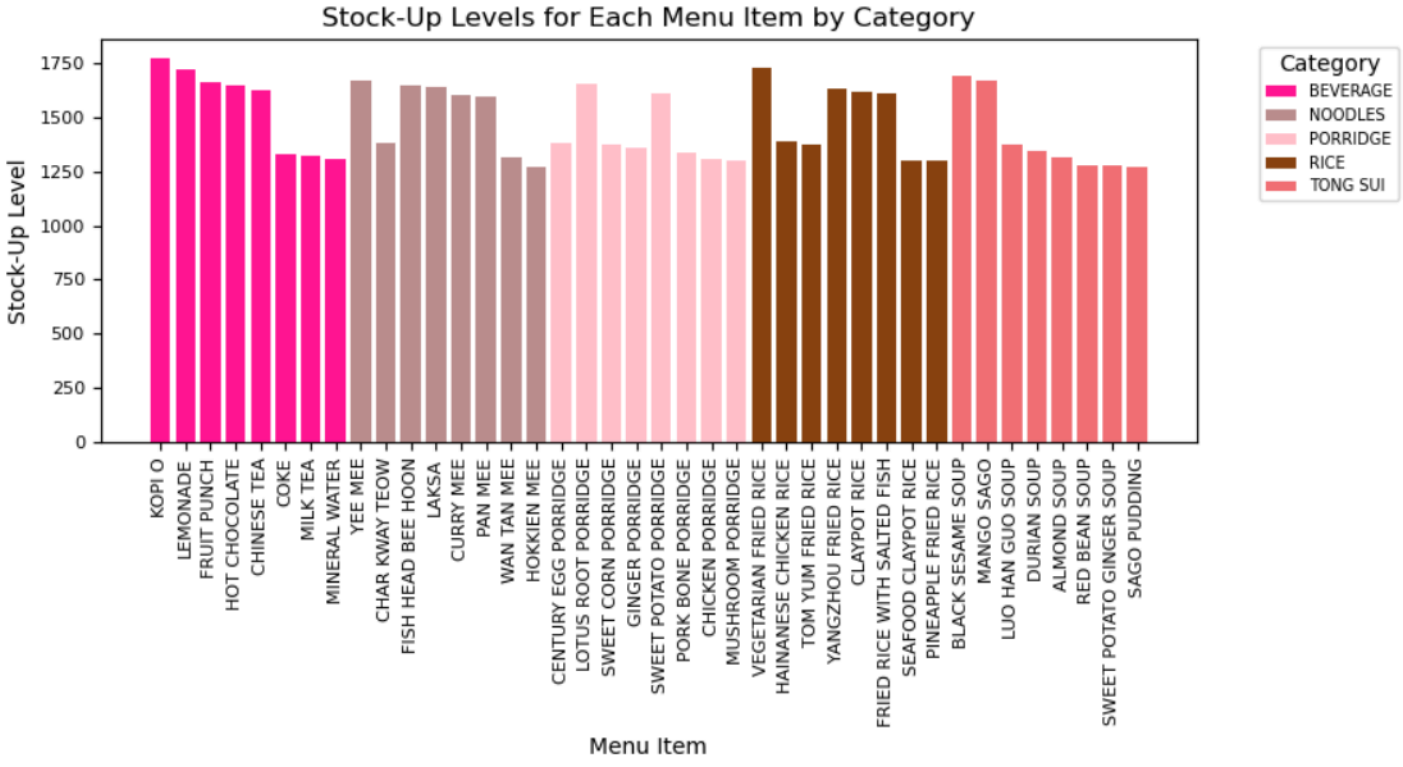

Sales by Quarter for BEVERAGE
Q1
Q2
Q3
Q4
Sales
0
200
400
600
800
1000
1200
1400
1600
KOPI O
LEMONADE
FRUIT PUNCH
HOT CHOCOLATE
CHINESE TEA
COKE
MILK TEA
MINERAL WATER
Menu Name


Sales by Quarter for NOODLES
Q1
Q2
Q3
Q4
Sales
0
200
400
600
800
1000
1200
1400
1600
YEE MEE
CHAR KWAY TEOW
FISH HEAD BEE HOON
LAKSA
CURRY MEE
PAN MEE
WAN TAN MEE
HOKKIEN MEE
Menu Name


Sales by Quarter for PORRIDGE
Sales
Menu Name
Q1
Q2
Q3
Q4
0
200
400
600
800
1000
1200
1400
CENTURY EGG PORRIDGE
LOTUS ROOT PORRIDGE
SWEET CORN PORRIDGE
GINGER PORRIDGE
SWEET POTATO PORRIDGE
PORK BONE PORRIDGE
CHICKEN PORRIDGE
MUSHROOM PORRIDGE


Sales by Quarter for RICE
Sales
Menu Name
Q1
Q2
Q3
Q4
0
200
400
600
800
1000
1200
1400
1600
VEGETARIAN FRIED RICE
HAINANESE CHICKEN RICE
TOM YUM FRIED RICE
YANGZHOU FRIED RICE
CLAYPOT RICE
FRIED RICE WITH SALTED FISH
SEAFOOD CLAYPOT RICE
PINEAPPLE FRIED RICE


Sales by Quarter for TONG SUI
Sales
Menu Name
Q1
Q2
Q3
Q4
0
200
400
600
800
1000
1200
1400
1600
BLACK SESAME SOUP
MANGO SAGO
LUO HAN GUO SOUP
DURIAN SOUP
ALMOND SOUP
RED BEAN SOUP
SWEET POTATO GINGER SOUP
SAGO PUDDING


Stock-Up Levels for Each Menu Item by Category
Stock-Up Level
Menu Item
Category
BEVERAGE
NOODLES
PORRIDGE
RICE
TONG SUI
0
250
500
750
1000
1250
1500
1750
KOPI O
LEMONADE
FRUIT PUNCH
HOT CHOCOLATE
CHINESE TEA
COKE
MILK TEA
MINERAL WATER
YEE MEE
CHAR KWAY TEOW
FISH HEAD BEE HOON
LAKSA
CURRY MEE
PAN MEE
WAN TAN MEE
HOKKIEN MEE
CENTURY EGG PORRIDGE
LOTUS ROOT PORRIDGE
SWEET CORN PORRIDGE
GINGER PORRIDGE
SWEET POTATO PORRIDGE
PORK BONE PORRIDGE
CHICKEN PORRIDGE
MUSHROOM PORRIDGE
VEGETARIAN FRIED RICE
HAINANESE CHICKEN RICE
TOM YUM FRIED RICE
YANGZHOU FRIED RICE
CLAYPOT RICE
FRIED RICE WITH SALTED FISH
SEAFOOD CLAYPOT RICE
PINEAPPLE FRIED RICE
BLACK SESAME SOUP
MANGO SAGO
LUO HAN GUO SOUP
DURIAN SOUP
ALMOND SOUP
RED BEAN SOUP
SWEET POTATO GINGER SOUP
SAGO PUDDING

This report analyzes the total quantity sold for each menu item by comparing sales data across the quarters in 2023. The goal is to identify trends and inconsistencies in sales performance and use this insight for smarter inventory stock-up decisions. By understanding sales patterns, we can adjust stock levels to avoid overstocking or shortages, ensuring inventory matches actual demand. The analysis reveals that sales fluctuate for each menu item, with no consistent pattern throughout the year. Some items experience a drop in certain periods, while others see a rise. For instance, Chinese Tea sales were inconsistent across the quarters. This data alone isn't enough for decision-making, so it's beneficial to compare sales between 2022 and 2023. Based on the percentage change, if a menu item sees a decline in sales compared to the previous year, it’s recommended to use the average quantity sold per quarter to guide stocking decisions. Conversely, if sales increase, the stock-up quantity should be raised by 20% to meet growing demand. For example, MILK TEA sales dropped by 7.53% from 2022 to 2023, so rather than increasing stock, using the average quarterly sales for inventory planning would be a more efficient approach.

3.3.2 Customer sales metrics in Selangor from year 2021 -2023 to determine new branch

SQL Code:

```
-- Terminal Settings
SET PAGESIZE 35
SET LINESIZE 120
SET VERIFY OFF

-- Title for Total Revenue Sales View
TTITLE ON
TTITLE CENTER
'+---------------------------------------------------------------
----------------------------+' SKIP 1 -
       CENTER '|                     CUSTOMER SALES METRICS IN
SELANGOR FROM YEAR 2021-2023                |' SKIP 1 -
       CENTER
'+---------------------------------------------------------------
----------------------------+' SKIP 2 -
       LEFT 'DATE: ' _DATE SKIP 1 -
       LEFT 'PAGE: ' FORMAT 999 SQL.PNO SKIP 2 -

-- Column Formatting
COLUMN "CUSTCITY" HEADING "CUSTOMER CITY" FORMAT A17
COLUMN "NUM_CUSTOMERS" HEADING "NUM|CUSTOMERS" FORMAT 9999
COLUMN "NUM_ORDERS" HEADING "NUM|ORDERS" FORMAT 9999
COLUMN "TOTALSALES" HEADING "TOTAL|SALES" FORMAT 999999.99
COLUMN "SALES_2021" HEADING "SALES|2021" FORMAT 999999.99
COLUMN "SALES_2022" HEADING "SALES|2022" FORMAT 999999.99
COLUMN "SALES_2023" HEADING "SALES|2023" FORMAT 999999.99
COLUMN "AVERAGE_SALES_3_YEARS" HEADING "AVG SALES| (3YRS)"
FORMAT 999999.99
COLUMN "AVG_SPEND_PER_CUSTOMER" HEADING "AVG SPEND|PER CUST"
FORMAT 999999.99
COLUMN "PERCENT_OF_TOTAL_SALES" HEADING "SALES| (%)" FORMAT
999.99

--QUERY
CREATE OR REPLACE VIEW TOTAL_REVENUE_SALE AS
SELECT
    CD.CUSTCITY AS "CUSTCITY",
    COUNT(DISTINCT CD.CUSTID) AS "NUM_CUSTOMERS",
    COUNT(SF.ORDERID) AS "NUM_ORDERS",
    SUM(SF.LINETOTAL) AS "TOTALSALES",
    SUM(CASE WHEN TO_CHAR(DD.CAL_YEAR = 2021 THEN SF.LINETOTAL
ELSE 0 END) AS "SALES_2021",
```

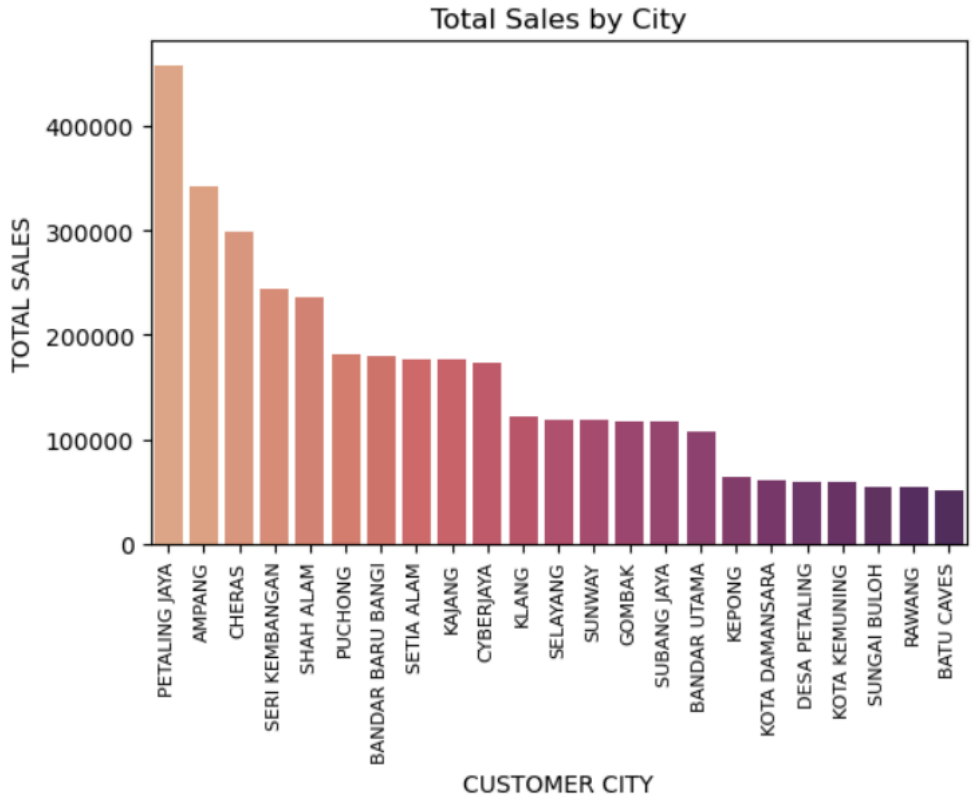
```
    SUM(CASE WHEN TO_CHAR(DD.CAL_YEAR = 2022 THEN SF.LINETOTAL
ELSE 0 END) AS "SALES_2022",
    SUM(CASE WHEN TO_CHAR(DD.CAL_YEAR = 2023 THEN SF.LINETOTAL
ELSE 0 END) AS "SALES_2023",
    SUM(SF.LINETOTAL) / 3 AS "AVERAGE_SALES_3_YEARS",
    (SUM(SF.LINETOTAL) / SUM(SUM(SF.LINETOTAL)) OVER()) * 100 AS
"PERCENT_OF_TOTAL_SALES",
    SUM(SF.LINETOTAL) / COUNT(DISTINCT CD.CUSTID) AS
"AVG_SPEND_PER_CUSTOMER"
FROM SALESFACT SF
JOIN CUSTOMER_DIM CD ON SF.CUSTOMER_KEY = CD.CUSTOMER_KEY
JOIN DATE_DIM DD ON SF.DATE_KEY = DD.DATE_KEY
WHERE CD.CUSTSTATE = 'SELANGOR'
      AND CD.ISCURRENT = 'Y'
      AND DD.CAL_DATE BETWEEN TO_DATE('01/01/2021',
'DD/MM/YYYY')
      AND TO_DATE('31/12/2023', 'DD/MM/YYYY')
GROUP BY CD.CUSTCITY
ORDER BY TOTALSALES DESC;

-- Select from the view to show results
SELECT * FROM TOTAL_REVENUE_SALE;

CLEAR COLUMNS
TTITLE OFF
```

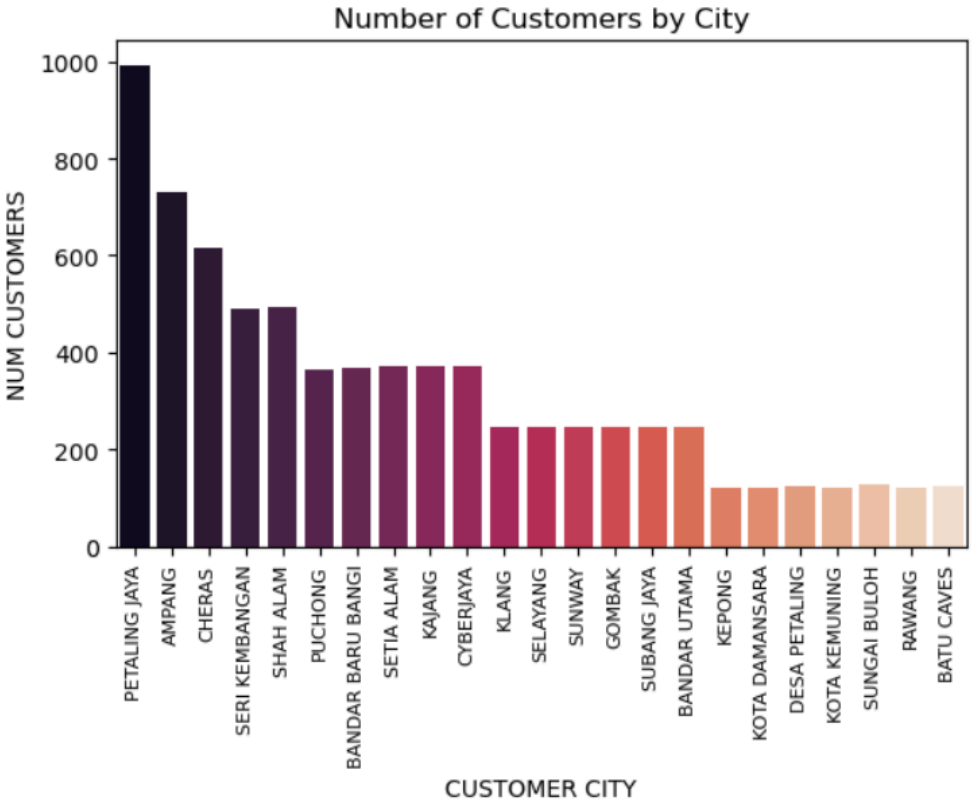
Output :

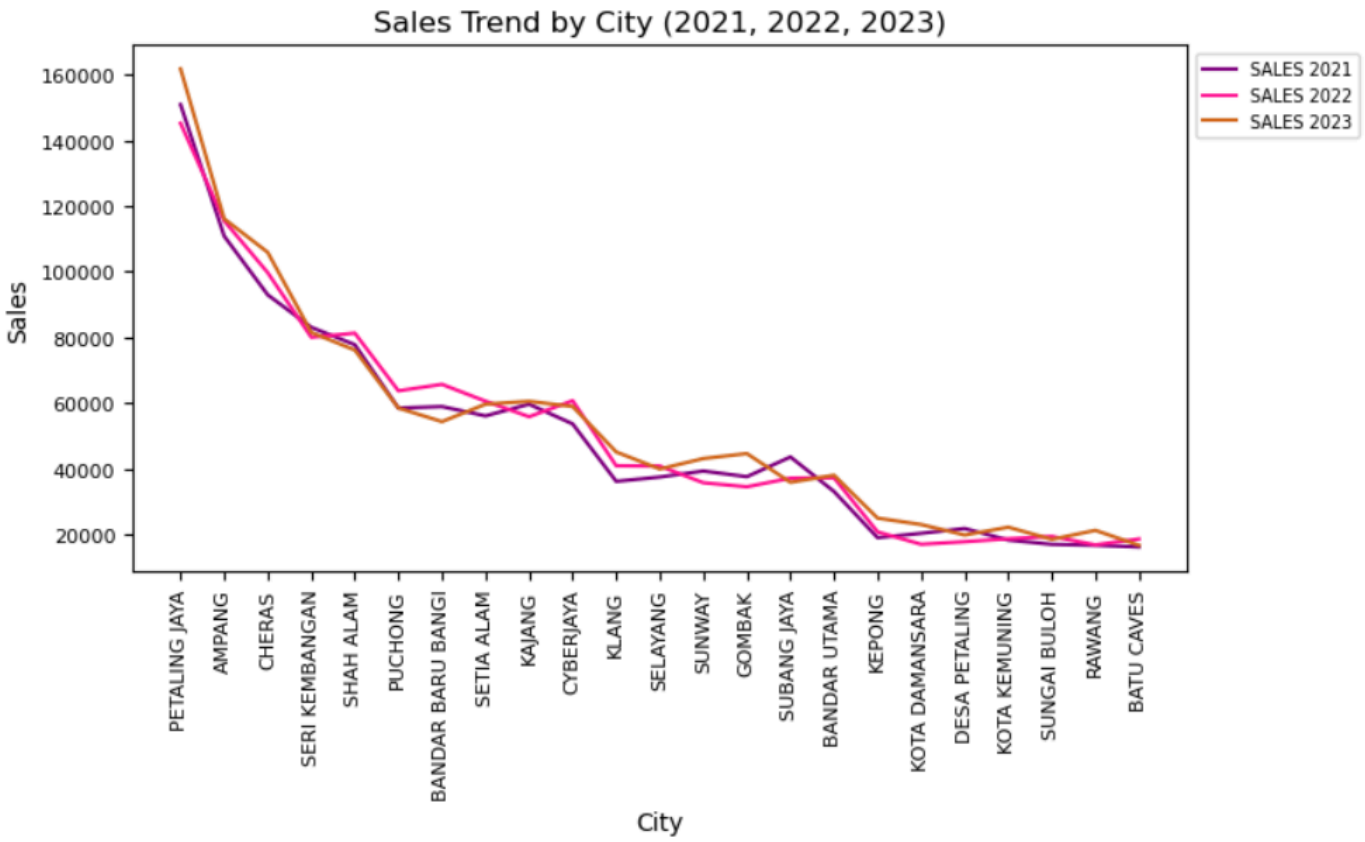
CUSTOMER SALES METRICS IN SELANGOR FROM YEAR 2021-2023

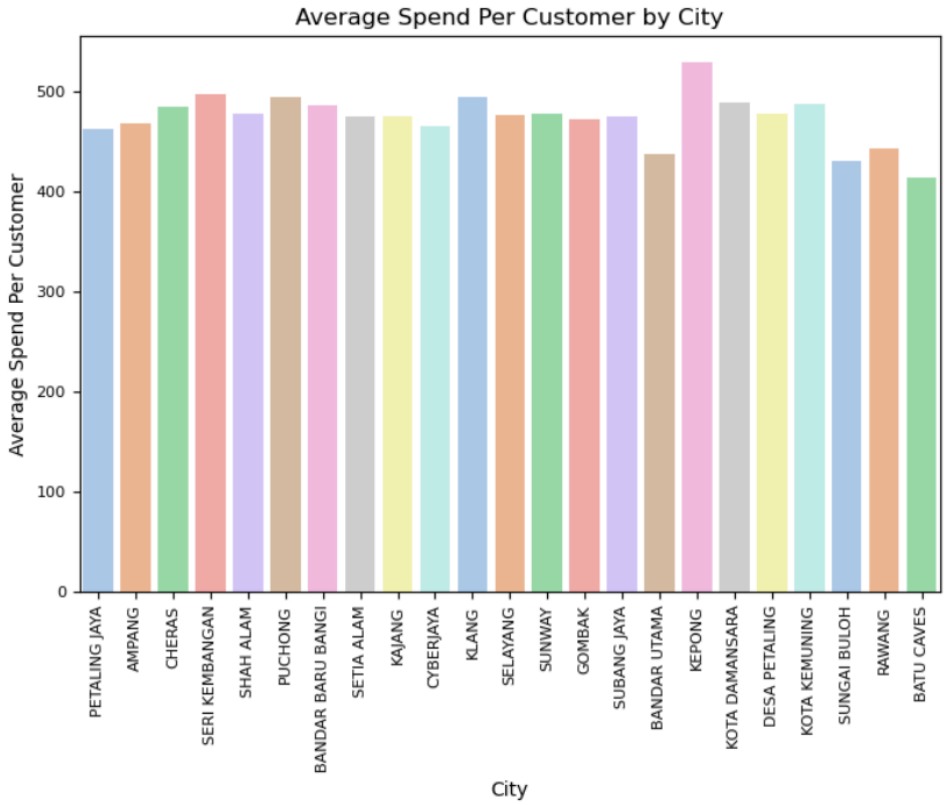
DATE: 19-SEP-24
PAGE: 1

CUSTOMER CITY	NUM CUSTOMERS	NUM ORDERS	TOTAL SALES	SALES 2021	SALES 2022	SALES 2023	AVG SALES (3YRS)	SALES (%)	AVG SPEND PER CUST
PETALING JAYA	992	9262	457514.78	150414.38	145263.18	161837.22	152504.93	12.83	461.20
AMPANG	732	7020	342555.92	110870.14	115542.65	116143.13	114185.31	9.61	467.97
CHERAS	617	6113	299138.81	93489.64	99732.14	105917.03	99712.94	8.39	484.83
SERI KEMBANGAN	492	4955	244580.54	83048.99	80033.29	81498.26	81526.85	6.86	497.11
SHAH ALAM	493	4644	235143.91	77747.59	81271.98	76124.34	78381.30	6.60	476.97
PUCHONG	366	3683	180557.11	58434.32	63681.86	58440.93	60185.70	5.06	493.33
BANDAR BARU BANGI	368	3550	178832.23	58882.55	65662.27	54287.41	59610.74	5.02	485.96
SETIA ALAM	371	3555	176173.38	56041.02	60514.95	59617.41	58724.46	4.94	474.86
KAJANG	371	3660	175888.09	59612.26	55791.29	60484.54	58629.36	4.93	474.09
CYBERJAYA	373	3578	173186.56	53599.50	60668.80	58918.26	57728.85	4.86	464.31
KLANG	247	2475	121951.36	36075.92	40833.87	45041.57	40650.45	3.42	493.73
SELAYANG	248	2422	117981.75	37397.78	40740.59	39843.38	39327.25	3.31	475.73
SUNWAY	247	2418	117918.08	39224.51	35652.45	43041.12	39306.03	3.31	477.40
GOMBAK	247	2435	116473.02	37499.76	34420.64	44552.62	38824.34	3.27	471.55
SUBANG JAYA	245	2322	116354.46	43524.01	37045.05	35785.40	38784.82	3.26	474.92
BANDAR UTAMA	247	2214	108100.71	32990.58	37156.73	37953.40	36033.57	3.03	437.65
KEPONG	122	1296	64425.17	18895.32	20660.43	24869.42	21475.06	1.81	528.08
KOTA DAMANSARA	123	1220	60041.53	20234.24	16893.56	22913.73	20013.84	1.68	488.14
DESA PETALING	124	1211	59091.51	21700.86	17674.25	19716.40	19697.17	1.66	476.54
KOTA KEMUNING	121	1151	58840.96	18172.99	18566.15	22101.82	19613.65	1.65	486.29
SUNGAI BULOH	127	1139	54550.26	16876.83	19279.73	18393.70	18183.42	1.53	429.53
RAWANG	123	1114	54437.52	16588.23	16706.30	21142.99	18145.84	1.53	442.58
BATU CAVES	124	1040	51176.88	16076.54	18510.89	16589.45	17058.96	1.44	412.72


Total Sales by City
TOTAL SALES
0
100000
200000
300000
400000
PETALING JAYA
AMPANG
CHERAS
SERI KEMBANGAN
SHAH ALAM
PUCHONG
BANDAR BARU BANGI
SETIA ALAM
KAJANG
CYBERJAYA
KLANG
SELAYANG
SUNWAY
GOMBAK
SUBANG JAYA
BANDAR UTAMA
KEPONG
KOTA DAMANSARA
DESA PETALING
KOTA KEMUNING
SUNGAI BULOH
RAWANG
BATU CAVES
CUSTOMER CITY


Number of Customers by City
NUM CUSTOMERS
0
200
400
600
800
1000
PETALING JAYA
AMPANG
CHERAS
SERI KEMBANGAN
SHAH ALAM
PUCHONG
BANDAR BARU BANGI
SETIA ALAM
KAJANG
CYBERJAYA
KLANG
SELAYANG
SUNWAY
GOMBAK
SUBANG JAYA
BANDAR UTAMA
KEPONG
KOTA DAMANSARA
DESA PETALING
KOTA KEMUNING
SUNGAI BULOH
RAWANG
BATU CAVES
CUSTOMER CITY


Sales Trend by City (2021, 2022, 2023)
SALES 2021
SALES 2022
SALES 2023
Sales
160000
140000
120000
100000
80000
60000
40000
20000
PETALING JAYA
AMPANG
CHERAS
SERI KEMBANGAN
SHAH ALAM
PUCHONG
BANDAR BARU BANGI
SETIA ALAM
KAJANG
CYBERJAYA
KLANG
SELAYANG
SUNWAY
GOMBAK
SUBANG JAYA
BANDAR UTAMA
KEPONG
KOTA DAMANSARA
DESA PETALING
KOTA KEMUNING
SUNGAI BULOH
RAWANG
BATU CAVES
City


Average Spend Per Customer by City
Average Spend Per Customer
500
400
300
200
100
0
PETALING JAYA
AMPANG
CHERAS
SERI KEMBANGAN
SHAH ALAM
PUCHONG
BANDAR BARU BANGI
SETIA ALAM
KAJANG
CYBERJAYA
KLANG
SELAYANG
SUNWAY
GOMBAK
SUBANG JAYA
BANDAR UTAMA
KEPONG
KOTA DAMANSARA
DESA PETALING
KOTA KEMUNING
SUNGAI BULOH
RAWANG
BATU CAVES
City

This report analyzes customer behavior and sales performance across various cities in Selangor from the year 2021 to 2023. By evaluating key metrics such as the number of customers, orders, total sales, and average spend per customer, the analysis reveals significant variations in sales across the cities. The data reveals significant sales variations across cities, with Petaling Jaya consistently leading in total sales. While cities like Ampang and Cheras show strong performance, they do not match the dominance of Petaling Jaya. Despite Kepong having the highest average spend per customer at RM 497.11, the data suggests that the city, along with others, has yet to fully capitalize on its potential for generating optimal revenue. Visualizations show that average spending per customer is relatively stable across cities, with only slight differences. Besides, Petaling Jaya stands out, accounting for 12.85% of total sales, with the highest number of customers at 992 and the most orders at 9,270. Given these strong insights, it is recommended that a new branch be strategically opened in Petaling Jaya. The city's dominant sales figures and substantial revenue contribution make it the ideal location for maximizing returns and leveraging its market potential. The detailed analysis and visualizations provide a clear foundation for this expansion decision, ensuring the new branch is well-positioned to capitalize on Petaling Jaya's robust market opportunities.

3.3.3 Menu analysis based on each category in Petaling Jaya from year 2021 to 2023

SQL Code:

```
-- Terminal Settings
SET PAGESIZE 38
SET LINESIZE 120
SET VERIFY OFF

-- Title for Sales Data
TTITLE ON
TTITLE LEFT '
+-------------------------------------------------------------
--------------------+' SKIP 1 -
       LEFT '    |                MENU ANALYSIS BASED ON EACH
CATEGORY IN PETALING JAYA               |' SKIP 1 -
       LEFT '
+-------------------------------------------------------------
--------------------+' SKIP 2 -
       LEFT   'DATE: ' _DATE SKIP 1 -
       LEFT 'PAGE: ' FORMAT 999 SQL.PNO SKIP 2 -

-- Column Formatting
BREAK ON CATEGORYNAME SKIP 1
COLUMN "CATEGORYNAME" HEADING "CATEGORY" FORMAT A10
COLUMN "MENU_RANK" HEADING "RANK" FORMAT 99
COLUMN "MENUNAME" HEADING "MENU ITEM" FORMAT A27
COLUMN "TOTAL_QUANTITY_SOLD" HEADING "TOTAL|SOLD" FORMAT 9999
COLUMN "QUANTITY_2021" HEADING "QTY|SOLD21" FORMAT 999
COLUMN "QUANTITY_2022" HEADING "QTY|SOLD22" FORMAT 999
COLUMN "QUANTITY_2023" HEADING "QTY|SOLD23" FORMAT 999
COLUMN "TOTAL_SALES" HEADING "TOTAL|SALES" FORMAT 99999.99
COLUMN "PERCENTAGE_OF_CATEGORY" HEADING
"CATEGORY|CONTRIBUTION|(%)" FORMAT 99.99

-- Create or Replace the View with Total Sales and Quantity Sold
Per Year
CREATE OR REPLACE VIEW MENU_SALES_VIEW AS
WITH CategorySales AS (
    SELECT M.CATEGORYNAME,
           SUM(F.QUANTITY) AS CATEGORY_TOTAL_QUANTITY
    FROM SALESFACT F
    JOIN DATE_DIM D ON F.DATE_KEY = D.DATE_KEY
    JOIN MENU_DIM M ON F.MENU_KEY = M.MENU_KEY
    JOIN CUSTOMER_DIM C ON F.CUSTOMER_KEY = C.CUSTOMER_KEY
    WHERE D.CAL_YEAR BETWEEN 2021 AND 2023
```

```
        AND C.CUSTCITY = 'PETALING JAYA'
    GROUP BY M.CATEGORYNAME
),
MenuSales AS (
    SELECT M.CATEGORYNAME,
           M.MENUNAME,
           SUM(F.QUANTITY) AS TOTAL_QUANTITY_SOLD,
           SUM(CASE WHEN D.CAL_YEAR = 2021 THEN F.QUANTITY ELSE
0 END) AS QUANTITY_2021,
           SUM(CASE WHEN D.CAL_YEAR = 2022 THEN F.QUANTITY ELSE
0 END) AS QUANTITY_2022,
           SUM(CASE WHEN D.CAL_YEAR = 2023 THEN F.QUANTITY ELSE
0 END) AS QUANTITY_2023,
           SUM(F.LINETOTAL) AS TOTAL_SALES
    FROM SALESFACT F
    JOIN DATE_DIM D ON F.DATE_KEY = D.DATE_KEY
    JOIN MENU_DIM M ON F.MENU_KEY = M.MENU_KEY
    JOIN CUSTOMER_DIM C ON F.CUSTOMER_KEY = C.CUSTOMER_KEY
    WHERE D.CAL_YEAR BETWEEN 2021 AND 2023 AND C.CUSTCITY =
'PETALING JAYA'
    GROUP BY M.CATEGORYNAME, M.MENUNAME
)
SELECT MS.CATEGORYNAME,
       ROW_NUMBER() OVER (PARTITION BY MS.CATEGORYNAME ORDER BY
MS.TOTAL_QUANTITY_SOLD DESC) AS MENU_RANK,
       MS.MENUNAME,
       MS.TOTAL_QUANTITY_SOLD,
       MS.QUANTITY_2021,
       MS.QUANTITY_2022,
       MS.QUANTITY_2023,
       MS.TOTAL_SALES,
      (MS.TOTAL_QUANTITY_SOLD / CS.CATEGORY_TOTAL_QUANTITY) *
100 AS PERCENTAGE_OF_CATEGORY
FROM MenuSales MS
JOIN CategorySales CS ON MS.CATEGORYNAME = CS.CATEGORYNAME
ORDER BY MS.CATEGORYNAME, MENU_RANK;


SELECT * FROM MENU_SALES_VIEW;
CLEAR BREAK
CLEAR COLUMNS
TTITLE OFF
```

Output:

MENU ANALYSIS BASED ON EACH CATEGORY IN PETALING JAYA

DATE: 19-SEP-24
PAGE: 1

CATEGORY	RANK	MENU ITEM	TOTAL SOLD	QTY SOLD21	QTY SOLD22	QTY SOLD23	TOTAL SALES	CONTRIBUTION (%)
BEVERAGE	1	CHINESE TEA	1478	636	364	478	1144.48	13.89
	2	HOT CHOCOLATE	1413	409	428	576	5469.00	13.28
	3	FRUIT PUNCH	1363	386	492	485	4450.43	12.81
	4	MILK TEA	1353	441	532	380	4523.98	12.72
	5	LEMONADE	1321	420	476	425	3800.55	12.42
	6	KOPI O	1267	419	403	445	4120.12	11.91
	7	MINERAL WATER	1263	440	417	406	1825.02	11.87
	8	COKE	1180	356	414	410	2809.95	11.09
NOODLES	1	FISH HEAD BEE HOON	1357	477	397	483	32847.50	13.01
	2	HOKKIEN MEE	1337	425	503	409	12823.50	12.82

	3	YEE MEE	1329	471	400	458	10122.40
12.74							
	4	PAN MEE	1328	403	476	449	15356.40
12.73							
	5	LAKSA	1327	425	492	410	25628.00
12.72							
	6	WAN TAN MEE	1318	458	424	436	10232.40
12.64							
	7	CURRY MEE	1255	474	350	431	14466.00
12.03							
	8	CHAR KWAY TEOW	1180	384	454	342	10189.35
11.31							
PORRIDGE	1	SWEET CORN PORRIDGE	1388	491	370	527	8000.10
13.48							
	2	CENTURY EGG PORRIDGE	1325	488	394	443	10292.40
12.87							
	3	PORK BONE PORRIDGE	1319	445	372	502	17776.50
12.81							
	4	CHICKEN PORRIDGE	1298	399	414	485	22683.60
12.61							
	5	LOTUS ROOT PORRIDGE	1276	435	421	420	9299.66
12.39							
	6	GINGER PORRIDGE	1252	377	458	417	7258.50
12.16							
	7	MUSHROOM PORRIDGE	1231	369	451	411	9542.00
11.96							
	8	SWEET POTATO PORRIDGE	1207	389	394	424	7005.30
11.72							

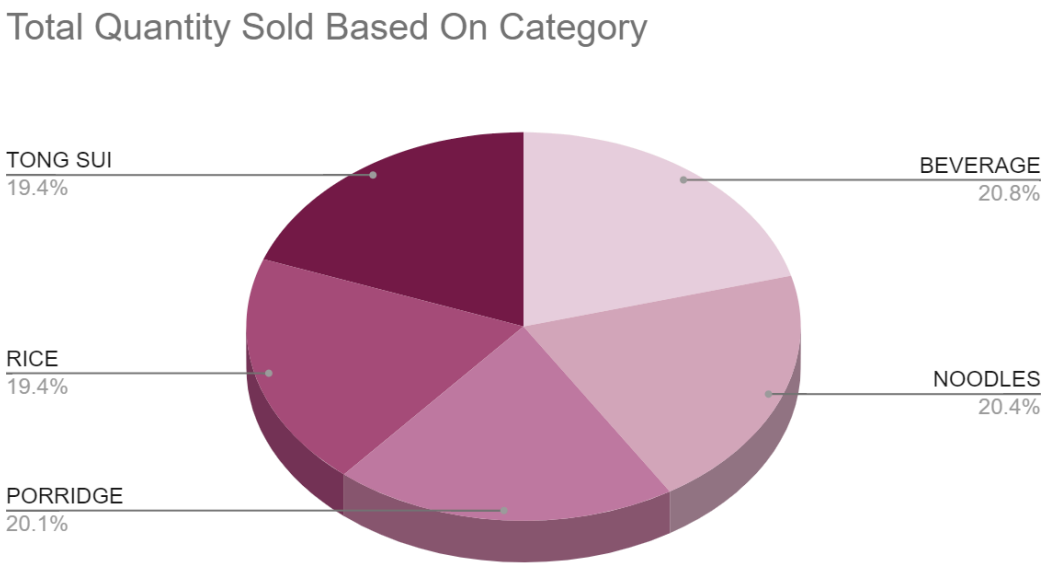
MENU ANALYSIS BASED ON EACH CATEGORY IN PETALING JAYA

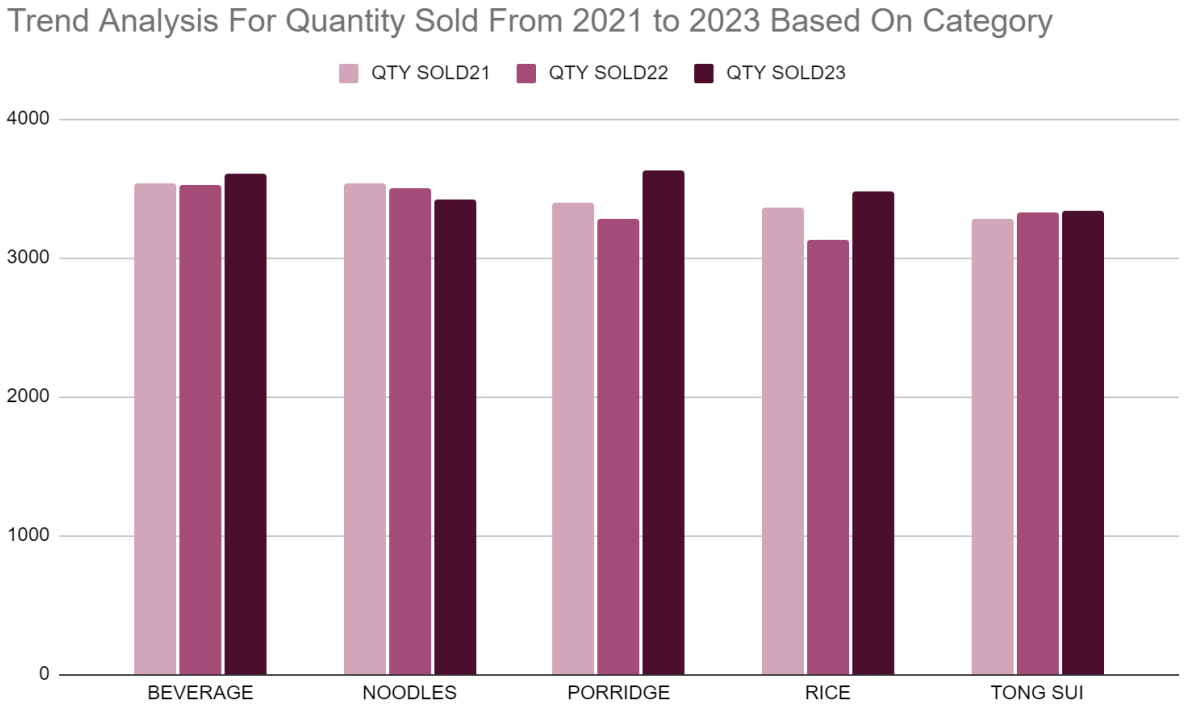
DATE: 19-SEP-24

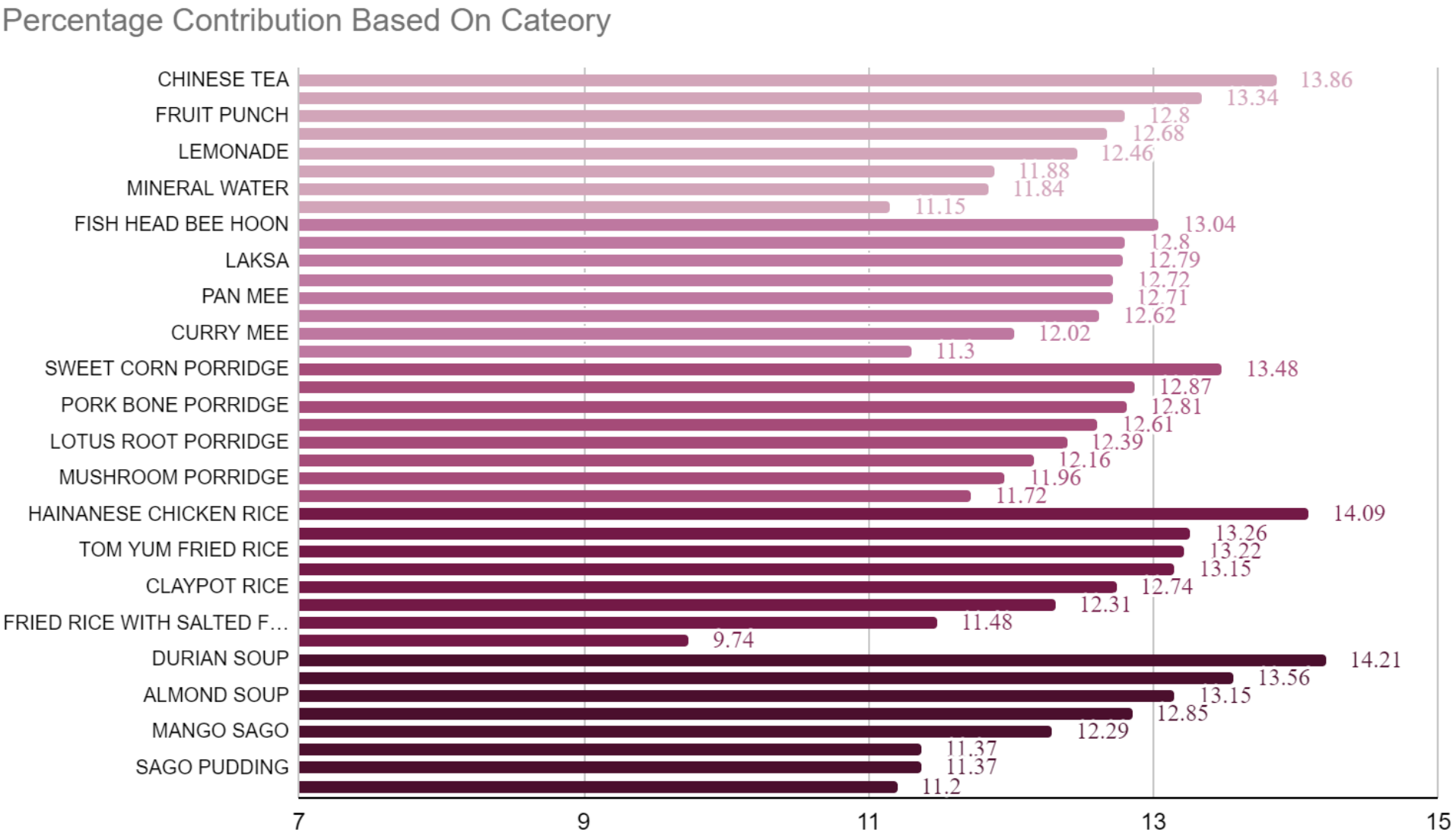
PAGE: 2

CATEGORY	RANK	MENU ITEM	TOTAL SOLD	QTY SOLD21	QTY SOLD22	QTY SOLD23	TOTAL SALES	CONTRIBUTION (%)
RICE	1	HAINANESE CHICKEN RICE	1403	442	486	475	13560.50	14.09
	2	PINEAPPLE FRIED RICE	1321	469	392	460	15331.80	13.27
	3	TOM YUM FRIED RICE	1317	448	455	414	17819.90	13.23
	4	VEGETARIAN FRIED RICE	1310	453	356	501	12731.50	13.16
	5	CLAYPOT RICE	1269	444	369	456	18429.00	12.75
	6	YANGZHOU FRIED RICE	1221	453	333	435	21117.60	12.27
	7	FRIED RICE WITH SALTED FISH	1143	342	354	447	13252.80	11.48
	8	SEAFOOD CLAYPOT RICE	970	303	378	289	14703.20	9.74
TONG SUI	1	DURIAN SOUP	1412	404	474	534	20489.25	14.22
	2	LUO HAN GUO SOUP	1340	431	466	443	6451.00	13.49
	3	ALMOND SOUP	1307	473	406	428	8737.05	13.16
	4	RED BEAN SOUP	1277	416	415	446	7461.00	12.86
	5	MANGO SAGO	1221	392	436	393	9493.60	12.30

```
             6 BLACK SESAME SOUP           1130    375    409    346   9756.45
11.38
             7 SAGO PUDDING                1130    400    335    395   9189.39
11.38
             8 SWEET POTATO GINGER SOUP    1113    382    381    350   7323.60
11.21
```


Total Quantity Sold Based On Category
TONG SUI
19.4%
BEVERAGE
20.8%
RICE
19.4%
NOODLES
20.4%
PORRIDGE
20.1%


Trend Analysis For Quantity Sold From 2021 to 2023 Based On Category
QTY SOLD21
QTY SOLD22
QTY SOLD23
4000
3000
2000
1000
0
BEVERAGE
NOODLES
PORRIDGE
RICE
TONG SUI


Percentage Contribution Based On Cateory
CHINESE TEA
FRUIT PUNCH
LEMONADE
MINERAL WATER
FISH HEAD BEE HOON
LAKSA
PAN MEE
CURRY MEE
SWEET CORN PORRIDGE
PORK BONE PORRIDGE
LOTUS ROOT PORRIDGE
MUSHROOM PORRIDGE
HAINANESE CHICKEN RICE
TOM YUM FRIED RICE
CLAYPOT RICE
FRIED RICE WITH SALTED F…
DURIAN SOUP
ALMOND SOUP
MANGO SAGO
SAGO PUDDING
13.86
13.34
12.8
12.68
12.46
11.88
11.84
11.15
13.04
12.8
12.79
12.72
12.71
12.62
12.02
11.3
13.48
12.87
12.81
12.61
12.39
12.16
11.96
11.72
14.09
13.26
13.22
13.15
12.74
12.31
11.48
9.74
14.21
13.56
13.15
12.85
12.29
11.37
11.37
11.2
7
9
11
13
15

This report analyzes menu performance in Petaling Jaya from 2021 to 2023, focusing on total sales, quantity sold, and the contribution of each menu item across categories such as Beverages, Noodles, Porridge, Rice, and Tong Sui. The analysis reveals that *Chinese Tea* (13.89%), *Hainanese Chicken Rice* (14.09%), and *Durian Soup* (14.22%) are the top-selling items, while *Seafood Claypot Rice* lags with a contribution of only 9.74%. Most categories show average contribution levels. *Tong Sui* has exhibited consistent growth over the past three years, while the trend for noodles has slightly declined. Based on these insights, it is recommended to prioritize marketing and promotions for top-performing items with a contribution of 13% or higher, ensuring their availability through optimized inventory management to avoid stock shortages. For lower-performing items like *Seafood Claypot Rice*, bundle deals or limited-time offers should be introduced to boost sales. This strategy not only drives revenue from high-demand items but also maximizes the sales potential of underperforming dishes through targeted inventory and promotional tactics, ensuring overall success at the new branch.

3.4 Wong Zi Ning

3.4.1 Top Menu Items by Promotion and Non-Promotion Sales Comparison for the Year-Month

SQL Code:

```
spool "C:\Users\user\Downloads\DWH\output1.txt"

set PAGESIZE 25
set LINESIZE 120

COLUMN NoPromo_SellQTY HEADING " No Promo | Selling Qty"
COLUMN NoPromo_Day_Count HEADING " No Promo | Day Count"
COLUMN Promo_SellQTY HEADING "Promo | Selling Qty"
COLUMN Promo_Day_Count HEADING "Promo | Day Count"
COLUMN NoPromo_SellQTY_PerDay HEADING " No Promo | Sell Qty |
Per Day"
COLUMN Promo_SellQTY_PerDay HEADING " Promo | Sell Qty | Per
Day"
COLUMN Ratio FORMAT A7

-- Prompt the user
ACCEPT yearmonth char prompt 'Enter the Year-Month (eg: 2024-4):
'

TTITLE ON
BTITLE ON

TTITLE                                                    CENTER
'=============================================================
=============' SKIP 1 -
         CENTER 'Top Menu Items by Promotion and Non-Promotion
Sales Comparison for ' &yearmonth SKIP 1 -
                                                          CENTER
'=============================================================
=============' SKIP 1 -
       LEFT 'Date Generated: ' _DATE -
       RIGHT 'Page ' SQL.PNO -
       SKIP 2-

BTITLE CENTER '-----End of Query-----'

CREATE OR REPLACE VIEW Menu_Sales_Comparison AS
WITH PromotionDays AS (
    SELECT
```

```
        COUNT(DISTINCT S.Date_Key) AS PromotionDays_Count
    FROM SalesFact S
    JOIN Promotion_dim P ON S.Promo_Key = P.Promo_Key
    WHERE P.Promo_Key != 1001
    AND S.Date_Key IN (
        SELECT Date_Key
        FROM Date_DIM
        WHERE cal_year_month = '&yearmonth'
    )
),
TotalDays AS (
    SELECT COUNT(DISTINCT Date_Key) AS TotalDays_Count
    FROM Date_DIM
    WHERE cal_year_month = '&yearmonth'
),
ItemSales AS (
    SELECT
        M.MenuName AS Menu_Name,
        D.Date_Key,
        SUM(S.Quantity) AS QTY,
        CASE
            WHEN S.Promo_Key != 1001 THEN 'Promotion'
            ELSE 'No Promotion'
        END AS PromoStatus
    FROM SalesFact S
    JOIN Promotion_dim P ON S.Promo_Key = P.Promo_Key
    JOIN Menu_dim M ON S.Menu_Key = M.Menu_Key
    JOIN Date_dim D ON S.Date_Key = D.Date_Key
    WHERE cal_year_month = '&yearmonth'
    GROUP BY M.MenuName, D.Date_Key,
    CASE
        WHEN S.Promo_Key != 1001 THEN 'Promotion'
        ELSE 'No Promotion'
    END
),
DailySales AS (
    SELECT
        Menu_Name,
        PromoStatus,
        SUM(QTY) AS Total_QTY
    FROM ItemSales
    GROUP BY Menu_Name, PromoStatus
),
SalesSummary AS (
    SELECT
```

```
        DS.Menu_Name,
          ROUND(SUM(CASE WHEN DS.PromoStatus = 'Promotion' THEN
DS.Total_QTY ELSE 0 END), 2) AS PromoQty,
        PD.PromotionDays_Count AS Promo_Day_Count,
         ROUND(SUM(CASE WHEN DS.PromoStatus = 'No Promotion' THEN
DS.Total_QTY ELSE 0 END), 2) AS NoPromoQty,
             (TD.TotalDays_Count - PD.PromotionDays_Count)  AS
NoPromo_Day_Count,
          ROUND(SUM(CASE WHEN DS.PromoStatus = 'Promotion' THEN
DS.Total_QTY / PD.PromotionDays_Count ELSE 0 END), 2) AS
PromoQty_Per_Day,
         ROUND(SUM(CASE WHEN DS.PromoStatus = 'No Promotion' THEN
DS.Total_QTY / (TD.TotalDays_Count - PD.PromotionDays_Count)
ELSE 0 END), 2) AS NoPromoQty_Per_Day,
          ROUND((SUM(CASE WHEN DS.PromoStatus = 'Promotion' THEN
DS.Total_QTY / PD.PromotionDays_Count ELSE 0 END) -
                  SUM(CASE WHEN DS.PromoStatus = 'No Promotion'
THEN     DS.Total_QTY     /          (TD.TotalDays_Count     -
PD.PromotionDays_Count) ELSE 0 END)), 2) AS QtyDifference
    FROM DailySales DS
    CROSS JOIN PromotionDays PD
    CROSS JOIN TotalDays TD
            GROUP   BY   DS.Menu_Name,   PD.PromotionDays_Count,
(TD.TotalDays_Count - PD.PromotionDays_Count)
),
Ranking AS (
    SELECT
        Menu_Name,
        PromoQty,
        Promo_Day_Count,
        PromoQty_Per_Day,
        NoPromoQty,
        NoPromo_Day_Count,
        NoPromoQty_Per_Day,
        QtyDifference,
        CASE
            WHEN NoPromoQty_Per_Day != 0 THEN
                        CONCAT('1: ', ROUND(PromoQty_Per_Day /
NoPromoQty_Per_Day, 2))
            ELSE
                NULL
        END AS Ratio,
        RANK() OVER (ORDER BY ROUND(
            CASE
                WHEN NoPromoQty_Per_Day != 0 THEN
```

```
                        (PromoQty_Per_Day / NoPromoQty_Per_Day *
100)
                ELSE
                    NULL
            END, 2) DESC) AS Rank
    FROM SalesSummary
)
SELECT
    Menu_Name,
    NoPromoQty AS NoPromo_SellQty,
    NoPromo_Day_Count,
    NoPromoQty_Per_Day AS NoPromo_SellQty_PerDay,
    PromoQty AS Promo_SellQty,
    Promo_Day_Count,
    PromoQty_Per_Day AS Promo_SellQty_PerDay,
    Ratio
FROM Ranking
WHERE Ratio IS NOT NULL
  AND ROUND(PromoQty_Per_Day / NoPromoQty_Per_Day, 2) > 1
ORDER BY RANK;

SELECT * FROM Menu_Sales_Comparison;

CLEAR BREAK;
TTITLE OFF;
BTITLE OFF;
SPOOL OFF;
```

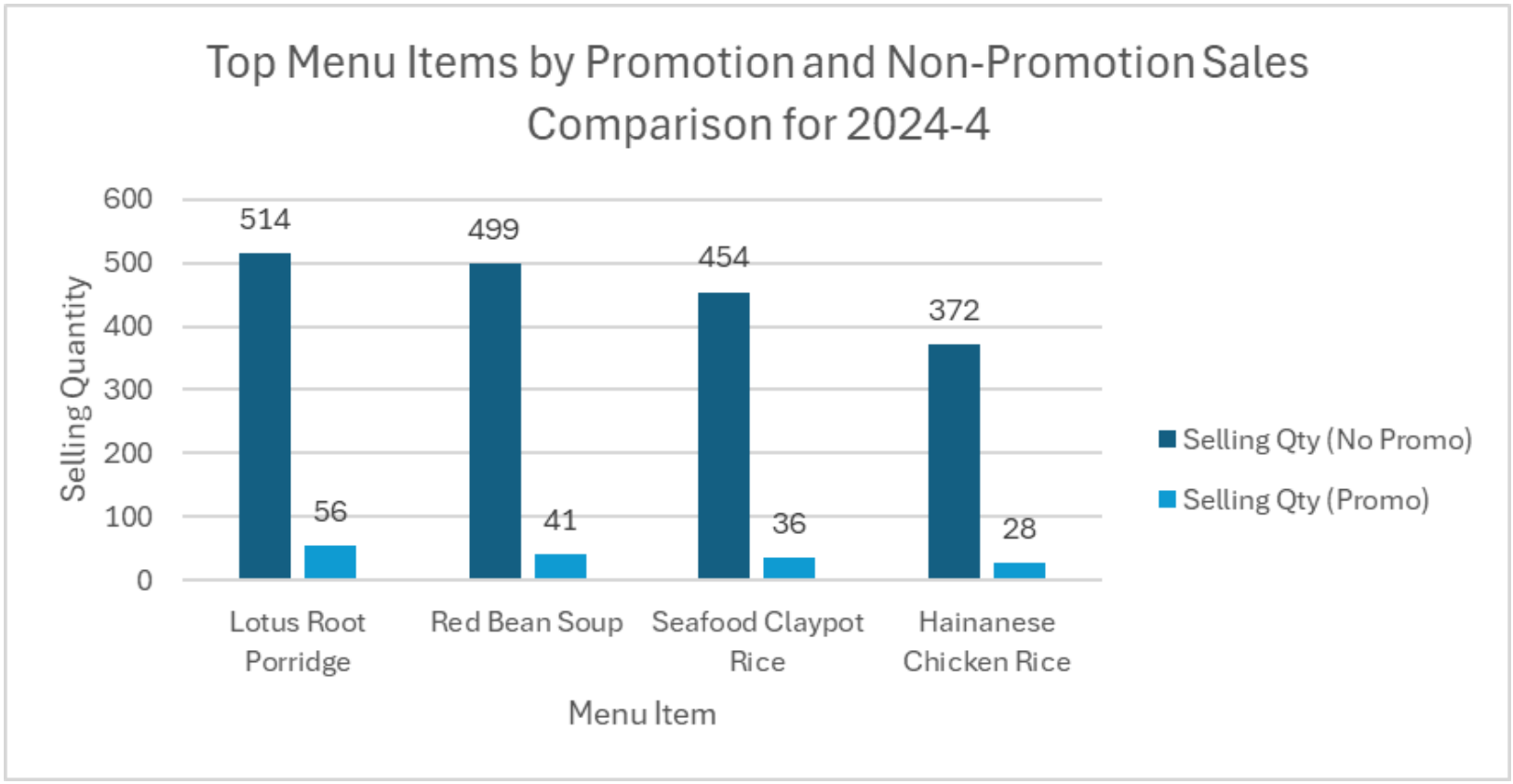
Output:

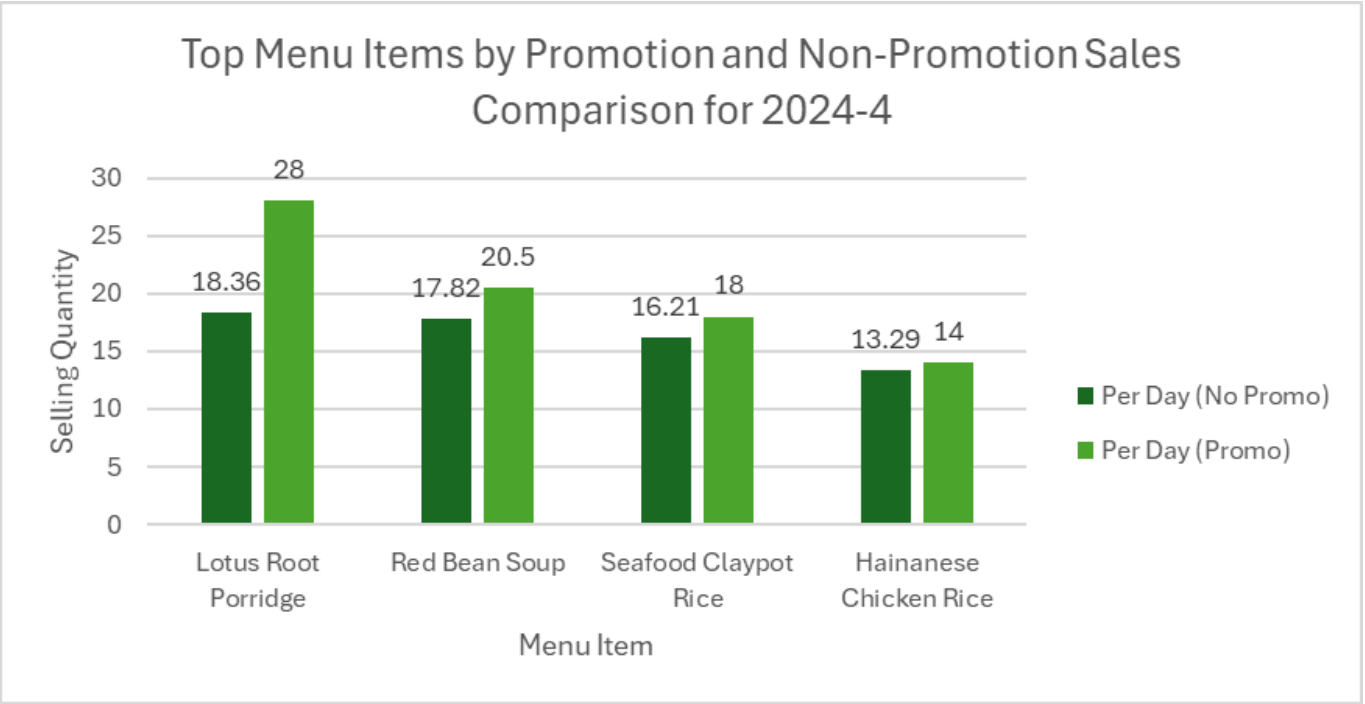
```
Enter the Year-Month (eg: 2024-4): 2024-4
old  11:         WHERE cal_year_month = '&yearmonth'
new  11:         WHERE cal_year_month = '2024-4'
old  17:     WHERE cal_year_month = '&yearmonth'
new  17:     WHERE cal_year_month = '2024-4'
old  32:     WHERE cal_year_month = '&yearmonth'
new  32:     WHERE cal_year_month = '2024-4'

View created.
```

```
                    ================================================================================
                      Top Menu Items by Promotion and Non-Promotion Sales Comparison for 2024-4
                    ================================================================================
Date Generated: 21-SEP-24                                                                    Page          1
```

MENU_NAME	No Promo Selling Qty	No Promo Day Count	No Promo Sell Qty Per Day	Promo Selling Qty	Promo Day Count	Promo Sell Qty Per Day	RATIO
LOTUS ROOT PORRIDGE	514	28	18.36	56	2	28	1: 1.53
RED BEAN SOUP	499	28	17.82	41	2	20.5	1: 1.15
SEAFOOD CLAYPOT RICE	454	28	16.21	36	2	18	1: 1.11
HAINANESE CHICKEN RICE	372	28	13.29	28	2	14	1: 1.05

```
                                      -----End of Query-----
```


Top Menu Items by Promotion and Non-Promotion Sales Comparison for 2024-4
Selling Quantity
600
500
400
300
200
100
0
514
56
499
41
454
36
372
28
Lotus Root Porridge
Red Bean Soup
Seafood Claypot Rice
Hainanese Chicken Rice
Menu Item
Selling Qty (No Promo)
Selling Qty (Promo)



The query compares menu item sales during promotional and non-promotional periods in April 2024 to assess the **effectiveness of promotions in boosting sales**. The analysis reveals that **Lotus Root Porridge** saw a significant increase in sales during promotions, with units sold rising from 18.36 per day to 28 per day—an **increase of 53%.** Similarly, other items like **Red Bean Soup** and **Seafood Claypot Rice** experienced sales boosts of 15% and 11%, respectively. These results indicate that promotions can effectively enhance sales for certain menu items.

Based on these findings, it is recommended to continue utilizing promotions, especially for items like Lotus Root Porridge and Red Bean Soup, particularly when their ingredients are approaching expiration. This strategy not only **drives higher sales** but also **helps in reducing ingredient wastage** by ensuring that perishable items are sold before they spoil. Thus, promotions can serve as an effective tool for both **boosting sales** and **optimizing inventory management.**

3.4.2 Annual Comparison of Dine-In and Delivery Orders with Sales and Growth Trends (2019-2023)

SQL Code:

```
spool "C:\Users\user\Downloads\DWH\output2.txt"

SET PAGESIZE 16
SET LINESIZE 120

TTITLE ON;
BTITLE ON;

COLUMN DELIVERY_ORDER_QTY HEADING " DELIVERY | ORDER QTY"
COLUMN DELIVERY_ORDER_SALES HEADING "DELIVERY | ORDER SALES"
FORMAT 999,999,999.99
COLUMN DELIVERY_SALES_PERCENT HEADING "DELIVERY | SALES PERCENT"
COLUMN DELIVERY_PERCENT HEADING "DELIVERY | QTY PERCENT"


COLUMN DINEIN_ORDER_QTY HEADING " DINE IN | ORDER QTY"
COLUMN DINEIN_ORDER_SALES HEADING "DINE IN | ORDER SALES" FORMAT
999,999,999.99
COLUMN DINEIN_SALES_PERCENT HEADING "DINE IN | SALES PERCENT"
COLUMN DINEIN_PERCENT HEADING "DINEIN | QTY PERCENT"


TTITLE CENTER
'================================================================
==============================' SKIP 1 -
       CENTER '  Annual Delivery vs. Dine-In Sales and Order
Comparison (2019-2023)  ' SKIP 1 -
       CENTER
'================================================================
==============================' SKIP 1 -
       LEFT 'Date Generated: ' _DATE -
       RIGHT 'Page ' SQL.PNO -
       SKIP 2 -

BTITLE CENTER '-----End of Query-----'

CREATE OR REPLACE VIEW Annual_Delivery_DineIn_Sales AS
WITH SalesCategory AS (
    SELECT
        D.Cal_Year,
        S.DELIVERYCOMPANYNAME,
```

```
        COUNT(DISTINCT S.OrderID) AS OrderCount,
        SUM(S.linetotal) AS TotalSales,
        CASE
            WHEN DELIVERYCOMPANYNAME = 'NO DELIVERY' THEN 'DINE
IN'
            ELSE 'DELIVERY'
        END AS Type
    FROM salesfact S
    JOIN Date_Dim D ON S.Date_key = D.Date_key
    WHERE D.cal_year BETWEEN 2019 AND 2023
    GROUP BY DELIVERYCOMPANYNAME, D.Cal_Year
),
TotalOrdersAndSalesPerYear AS (
    SELECT
        Cal_Year,
        SUM(OrderCount) AS TotalOrdersYear,
        SUM(TotalSales) AS TotalSalesYear
    FROM SalesCategory
    GROUP BY Cal_Year
)
SELECT
    SC.Cal_Year AS Year,
    SUM(CASE WHEN SC.Type = 'DELIVERY' THEN SC.OrderCount ELSE 0
END) AS DELIVERY_ORDER_QTY,
    SUM(CASE WHEN SC.Type = 'DINE IN' THEN SC.OrderCount ELSE 0
END) AS DINEIN_ORDER_QTY,
    ROUND(SUM(CASE WHEN SC.Type = 'DELIVERY' THEN SC.OrderCount
ELSE 0 END) * 100.0 / TOPY.TotalOrdersYear, 2) AS
DELIVERY_PERCENT,
    ROUND(SUM(CASE WHEN SC.Type = 'DINE IN' THEN SC.OrderCount
ELSE 0 END) * 100.0 / TOPY.TotalOrdersYear, 2) AS
DINEIN_PERCENT,
    SUM(CASE WHEN SC.Type = 'DELIVERY' THEN SC.TotalSales ELSE 0
END) AS DELIVERY_ORDER_SALES,
    SUM(CASE WHEN SC.Type = 'DINE IN' THEN SC.TotalSales ELSE 0
END) AS DINEIN_ORDER_SALES,
    ROUND(SUM(CASE WHEN SC.Type = 'DELIVERY' THEN SC.TotalSales
ELSE 0 END) * 100.0 / TOPY.TotalSalesYear, 2) AS
DELIVERY_SALES_PERCENT,
    ROUND(SUM(CASE WHEN SC.Type = 'DINE IN' THEN SC.TotalSales
ELSE 0 END) * 100.0 / TOPY.TotalSalesYear, 2) AS
DINEIN_SALES_PERCENT
FROM SalesCategory SC
JOIN TotalOrdersAndSalesPerYear TOPY ON SC.Cal_Year =
TOPY.Cal_Year
```

```
GROUP BY SC.Cal_Year, TOPY.TotalOrdersYear, TOPY.TotalSalesYear
ORDER BY SC.Cal_Year;

SELECT * FROM Annual_Delivery_DineIn_Sales;

CLEAR BREAK;
TTITLE OFF;
BTITLE OFF;
SPOOL OFF;
```

Output:

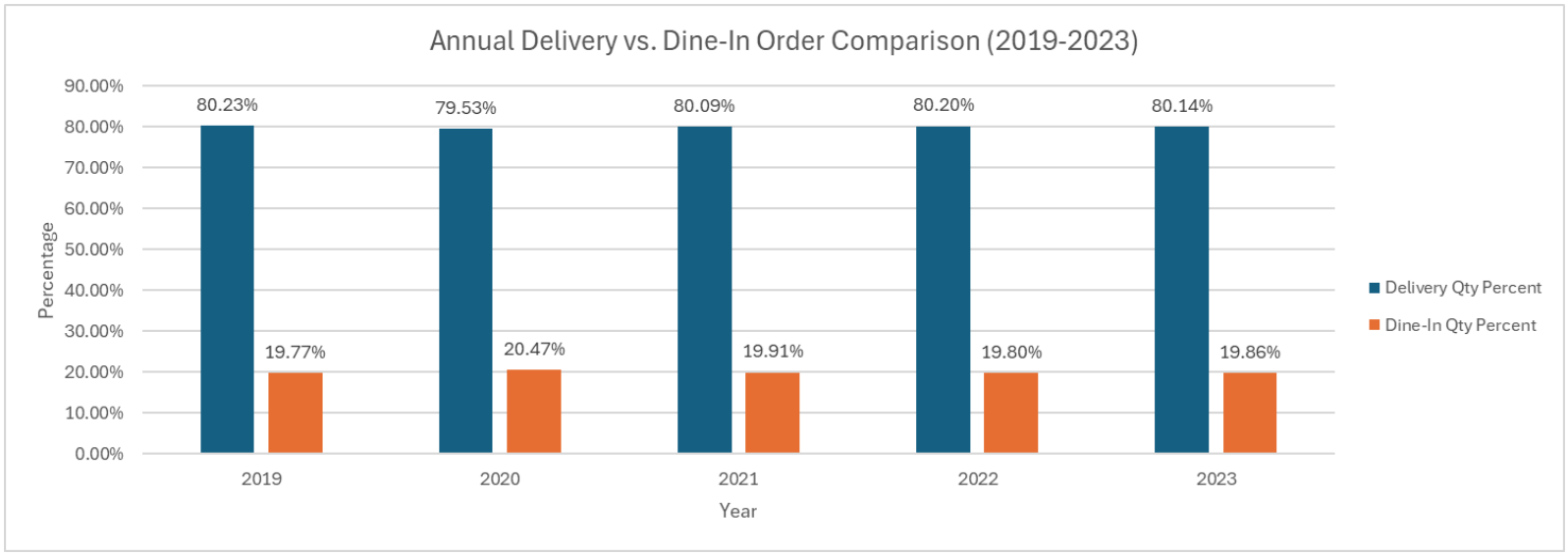
```
View created.
```

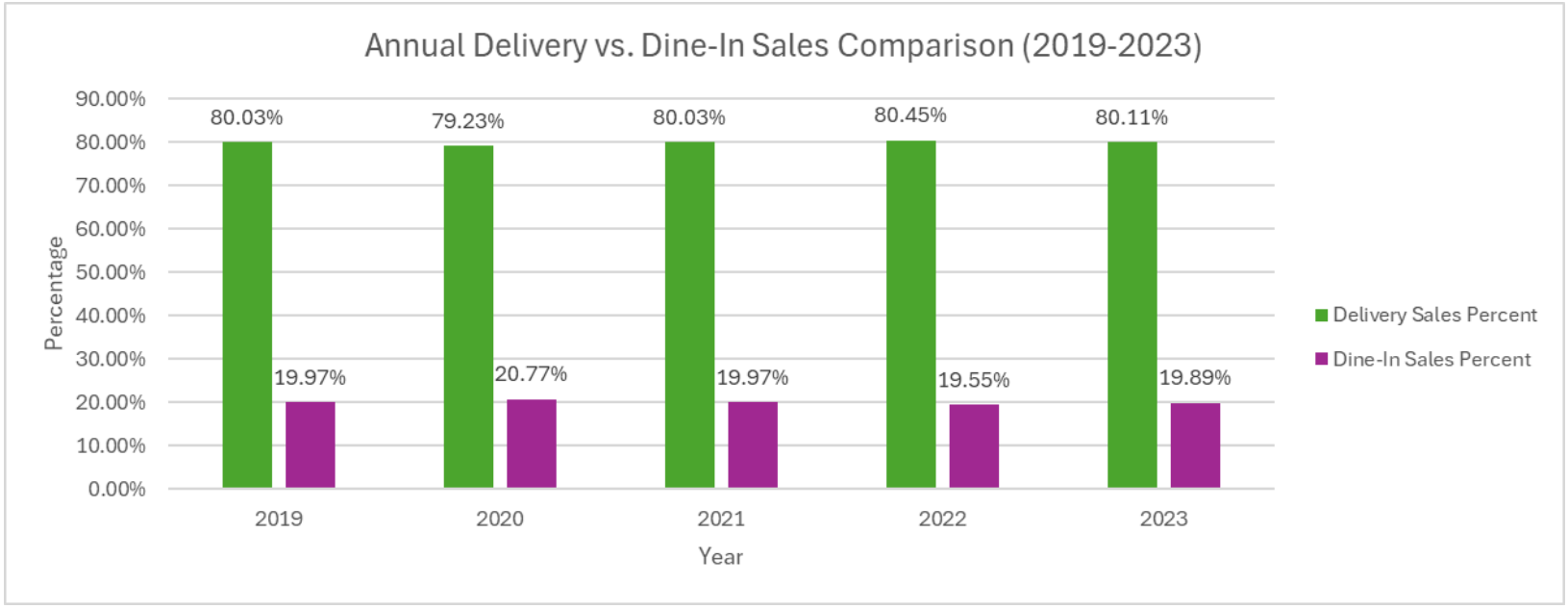
Annual Delivery vs. Dine-In Sales and Order Comparison (2019-2023)

Date Generated: 21-SEP-24 Page 1

YEAR	DELIVERY ORDER QTY	DINE IN ORDER QTY	DELIVERY QTY PERCENT	DINEIN QTY PERCENT	DELIVERY ORDER SALES	DINE IN ORDER SALES	DELIVERY SALES PERCENT	DINE IN SALES PERCENT
2019	10844	2672	80.23	19.77	1,577,472.19	393,714.52	80.03	19.97
2020	10853	2794	79.53	20.47	1,475,906.36	386,895.24	79.23	20.77
2021	11162	2775	80.09	19.91	1,566,743.26	390,944.35	80.03	19.97
2022	10850	2679	80.2	19.8	1,536,577.56	373,386.24	80.45	19.55
2023	11022	2732	80.14	19.86	1,630,632.61	404,870.57	80.11	19.89

-----End of Query-----


Annual Delivery vs. Dine-In Order Comparison (2019-2023)
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
Percentage
80.23%
19.77%
79.53%
20.47%
80.09%
19.91%
80.20%
19.80%
80.14%
19.86%
2019
2020
2021
2022
2023
Year
Delivery Qty Percent
Dine-In Qty Percent


Annual Delivery vs. Dine-In Sales Comparison (2019-2023)
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
Percentage
80.03%
19.97%
79.23%
20.77%
80.03%
19.97%
80.45%
19.55%
80.11%
19.89%
2019
2020
2021
2022
2023
Year
Delivery Sales Percent
Dine-In Sales Percent

The query results provide a comprehensive overview of delivery and dine-in order trends from 2019 to 2023, including key metrics like order quantities, sales figures, and percentage contributions for each type. Over these years, **delivery orders consistently represented a much larger share** of both total orders and sales compared to dine-in. Specifically, delivery orders accounted for **about 80% of both total orders and sales annually,** with delivery sales contributing between 79.23% and 80.45% of total revenue. This indicates a **strong and stable preference for delivery services.**

The purpose of analyzing this data is to determine **whether opening a new counter exclusively for delivery orders would be beneficial**. Given the steady high volume of delivery orders, **having a dedicated delivery counter** could improve operational efficiency and reduce congestion at the main counter. This would likely enhance the overall customer experience for both delivery and dine-in patrons

3.4.3 Daily and Weekly Peak Order and Sales Analysis by Time of Day in the Year

SQL Code:

```
spool "C:\Users\user\Downloads\DWH\output3.txt"

SET PAGESIZE 52
SET LINESIZE 100

BREAK ON DAY SKIP 1
COLUMN "PEAK OF DAY" FORMAT A10 HEADING "PEAK OF | THE DAY"
COLUMN "PEAK OF WEEK" FORMAT A10 HEADING "PEAK OF | THE WEEK"
COLUMN "AVG Orders Per Hour" HEADING "AVG ORDERS | PER HOUR"
COLUMN "AVG SALES" FORMAT 999,999.99
COMPUTE SUM LABEL "TOTAL" OF "AVG ORDERS" ON DAY
COMPUTE SUM LABEL "TOTAL" OF "AVG SALES" ON DAY

-- Prompt the user
ACCEPT year NUMBER prompt 'Enter the Year (eg: 2024): '

TTITLE ON
BTITLE ON

TTITLE                                                          CENTER
'================================================================
============' SKIP 1 -
       CENTER 'Daily and Weekly Peak Order and Sales Analysis by
Time of Day in ' &year SKIP 1 -
                                                                CENTER
'================================================================
============' SKIP 1 -
       LEFT 'Date Generated: ' _DATE -
       RIGHT 'Page ' SQL.PNO -
       SKIP 2 -

BTITLE CENTER '-----End of Query-----'

CREATE OR REPLACE VIEW Daily_Weekly_Peak_Order_Sales AS
WITH cte_time_of_day AS (
    SELECT
        d.day_of_week,
        CASE
            WHEN TO_NUMBER(SUBSTR(s.ordertime, 1, 2)) BETWEEN 10
AND 11 THEN 'Morning (10:00 - 11:59AM)'
            WHEN TO_NUMBER(SUBSTR(s.ordertime, 1, 2)) BETWEEN 12
AND 17 THEN 'Noon (12:00 - 5:59PM)'
```

```
            WHEN TO_NUMBER(SUBSTR(s.ordertime, 1, 2)) BETWEEN 18
AND 21 THEN 'Night (6:00 - 9:59PM)'
            ELSE 'Other'
        END AS time_period,
        d.date_key,
        SUM(s.linetotal) AS total_sales,
        COUNT(DISTINCT S.ORDERID) AS line_count
    FROM Salesfact S
    JOIN Date_Dim D ON S.date_key = D.Date_Key
    WHERE cal_Year = &year
    GROUP BY
        d.day_of_week,
        CASE
            WHEN TO_NUMBER(SUBSTR(s.ordertime, 1, 2)) BETWEEN 10
AND 11 THEN 'Morning (10:00 - 11:59AM)'
            WHEN TO_NUMBER(SUBSTR(s.ordertime, 1, 2)) BETWEEN 12
AND 17 THEN 'Noon (12:00 - 5:59PM)'
            WHEN TO_NUMBER(SUBSTR(s.ordertime, 1, 2)) BETWEEN 18
AND 21 THEN 'Night (6:00 - 9:59PM)'
            ELSE 'Other'
        END,
        d.date_key
),
daily_averages AS (
    SELECT
        day_of_week,
        time_period,
        AVG(total_sales) AS daily_avg_sales,
        AVG(line_count) AS daily_avg_orders
    FROM cte_time_of_day
    GROUP BY day_of_week, time_period
),
ranked_averages AS (
    SELECT
        day_of_week,
        time_period,
        AVG(daily_avg_sales) AS "AVG SALES",
        AVG(daily_avg_orders) AS "AVG ORDERS",
        CASE
            WHEN time_period = 'Morning (10:00 - 11:59AM)' THEN
AVG(daily_avg_orders) / 2
            WHEN time_period = 'Noon (12:00 - 5:59PM)' THEN
AVG(daily_avg_orders) / 6
            WHEN time_period = 'Night (6:00 - 9:59PM)' THEN
AVG(daily_avg_orders) / 4
```

```
            ELSE NULL
        END AS "AVG ORDERS PER HOUR",
              RANK() OVER (PARTITION BY day_of_week ORDER BY
AVG(daily_avg_orders) DESC) AS day_rank,
           RANK() OVER (ORDER BY AVG(daily_avg_orders) DESC) AS
week_rank
    FROM daily_averages
    GROUP BY day_of_week, time_period
),
day_peaks AS (
    SELECT
        day_of_week,
        MAX("AVG ORDERS PER HOUR") AS peak_of_day
    FROM ranked_averages
    GROUP BY day_of_week
),
week_peaks AS (
    SELECT
        MAX("AVG ORDERS PER HOUR") AS peak_of_week
    FROM ranked_averages
)
SELECT
    ra.day_of_week AS DAY,
    ra.time_period AS TIME,
    ROUND(ra."AVG ORDERS", 0) AS "AVG ORDERS",
    ROUND(ra."AVG SALES", 2) AS "AVG SALES",
    ROUND(ra."AVG ORDERS PER HOUR", 2) AS "AVG ORDERS PER HOUR",
    CASE
           WHEN ra."AVG ORDERS PER HOUR" = dp.peak_of_day THEN
'PEAK'
        ELSE NULL
    END AS "PEAK OF DAY",
    CASE
          WHEN ra."AVG ORDERS PER HOUR" = (SELECT peak_of_week
FROM week_peaks) THEN 'PEAK'
        ELSE NULL
    END AS "PEAK OF WEEK"
FROM ranked_averages ra
JOIN day_peaks dp
    ON ra.day_of_week = dp.day_of_week
ORDER BY ra.day_of_week,
         CASE
              WHEN ra.time_period = 'Morning (10:00 - 11:59AM)'
THEN 1
```

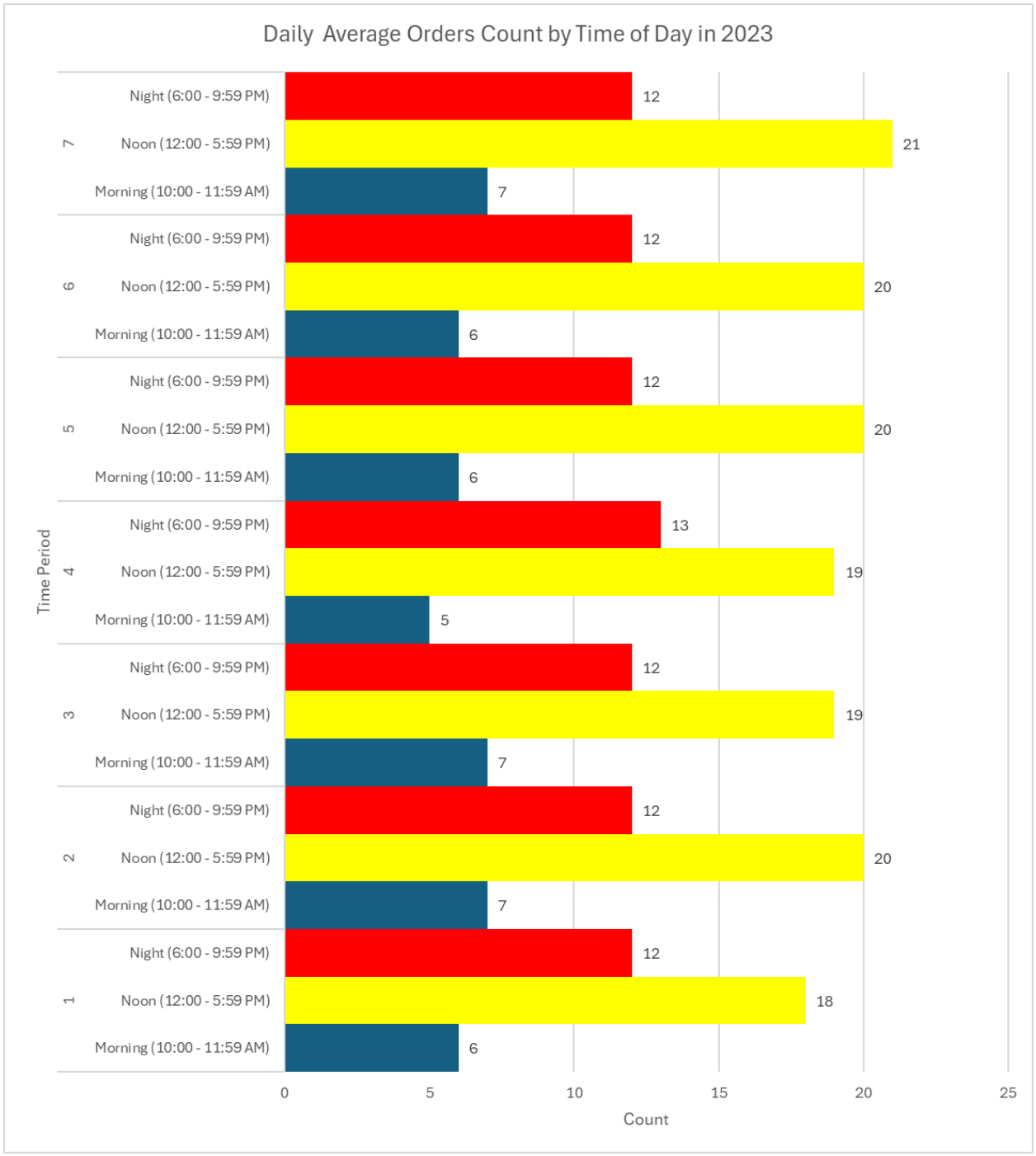
```
                WHEN ra.time_period = 'Noon (12:00 - 5:59PM)' THEN
2
                WHEN ra.time_period = 'Night (6:00 - 9:59PM)' THEN
3
              ELSE 4
          END;

SELECT *
FROM Daily_Weekly_Peak_Order_Sales;

CLEAR COLUMN;
CLEAR COMPUTE;
CLEAR BREAK;
TTITLE OFF;
BTITLE OFF;
SPOOL OFF;
```

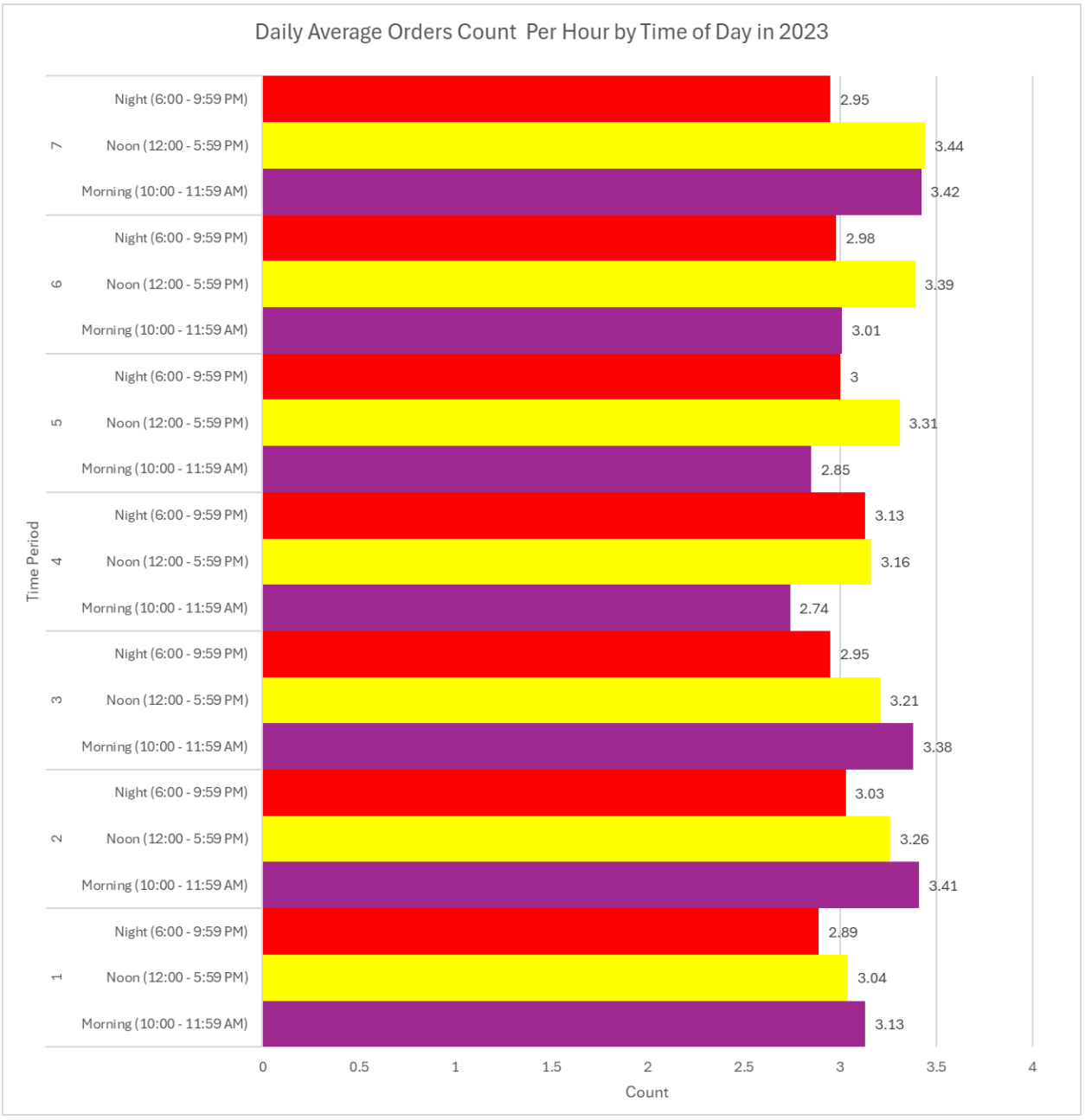
Output:

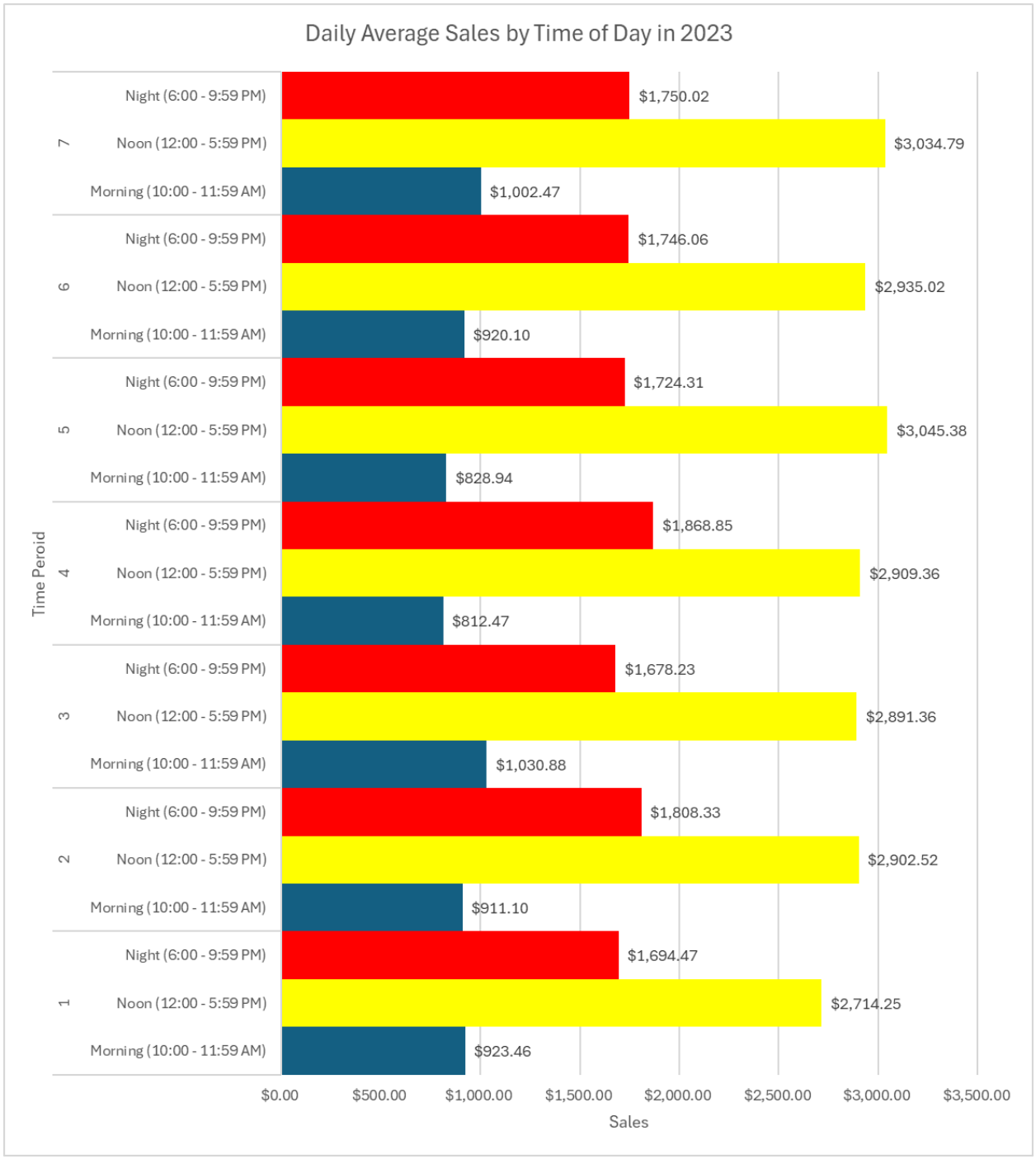
```
Enter the Year (eg: 2024): 2023
old  16:     WHERE cal_Year = &year
new  16:     WHERE cal_Year =       2023

View created.
```

```
            ================================================================================
                Daily and Weekly Peak Order and Sales Analysis by Time of Day in 2023
            ================================================================================
Date Generated: 21-SEP-24                                                          Page           1

                                                                AVG ORDERS  PEAK OF    PEAK OF
       DAY TIME                        AVG ORDERS   AVG SALES    PER HOUR  THE DAY    THE WEEK
---------- ------------------------- ---------- ----------- ----------- ---------- ----------
         1 Morning (10:00 - 11:59AM)          6      923.46        3.13 PEAK
           Noon (12:00 - 5:59PM)             18    2,714.25        3.04
           Night (6:00 - 9:59PM)             12    1,694.47        2.89
********** ------------------------- ---------- -----------
TOTAL                                        36    5,332.18

         2 Morning (10:00 - 11:59AM)          7      911.10        3.41 PEAK
           Noon (12:00 - 5:59PM)             20    2,902.52        3.26
           Night (6:00 - 9:59PM)             12    1,808.33        3.03
********** ------------------------- ---------- -----------
TOTAL                                        39    5,621.95

         3 Morning (10:00 - 11:59AM)          7    1,030.88        3.38 PEAK
           Noon (12:00 - 5:59PM)             19    2,891.36        3.21
           Night (6:00 - 9:59PM)             12    1,678.23        2.95
********** ------------------------- ---------- -----------
TOTAL                                        38    5,600.47

         4 Morning (10:00 - 11:59AM)          5      812.47        2.74
```

```
           Noon (12:00 - 5:59PM)             19    2,909.36        3.16 PEAK
           Night (6:00 - 9:59PM)             13    1,868.85        3.13
********** ------------------------------ ---------- ----------
TOTAL                                        37    5,590.68

         5 Morning (10:00 - 11:59AM)          6      828.94        2.85
           Noon (12:00 - 5:59PM)             20    3,045.38        3.31 PEAK
           Night (6:00 - 9:59PM)             12    1,724.31           3
********** ------------------------------ ---------- ----------
TOTAL                                        38    5,598.63

         6 Morning (10:00 - 11:59AM)          6      920.10        3.01
           Noon (12:00 - 5:59PM)             20    2,935.02        3.39 PEAK
           Night (6:00 - 9:59PM)             12    1,746.06        2.98
********** ------------------------------ ---------- ----------
TOTAL                                        38    5,601.18

         7 Morning (10:00 - 11:59AM)          7    1,002.47        3.42
           Noon (12:00 - 5:59PM)             21    3,034.79        3.44 PEAK       PEAK
           Night (6:00 - 9:59PM)             12    1,750.02        2.95
********** ------------------------------ ---------- ----------
TOTAL                                        40    5,787.28

                                      -----End of Query-----

21 rows selected.
```


Daily Average Orders Count by Time of Day in 2023
Time Period
Count
0
5
10
15
20
25
7
Night (6:00 - 9:59 PM) 12
Noon (12:00 - 5:59 PM) 21
Morning (10:00 - 11:59 AM) 7
6
Night (6:00 - 9:59 PM) 12
Noon (12:00 - 5:59 PM) 20
Morning (10:00 - 11:59 AM) 6
5
Night (6:00 - 9:59 PM) 12
Noon (12:00 - 5:59 PM) 20
Morning (10:00 - 11:59 AM) 6
4
Night (6:00 - 9:59 PM) 13
Noon (12:00 - 5:59 PM) 19
Morning (10:00 - 11:59 AM) 5
3
Night (6:00 - 9:59 PM) 12
Noon (12:00 - 5:59 PM) 19
Morning (10:00 - 11:59 AM) 7
2
Night (6:00 - 9:59 PM) 12
Noon (12:00 - 5:59 PM) 20
Morning (10:00 - 11:59 AM) 7
1
Night (6:00 - 9:59 PM) 12
Noon (12:00 - 5:59 PM) 18
Morning (10:00 - 11:59 AM) 6


Daily Average Orders Count Per Hour by Time of Day in 2023
Time Period
7
Night (6:00 - 9:59 PM)
Noon (12:00 - 5:59 PM)
Morning (10:00 - 11:59 AM)
6
Night (6:00 - 9:59 PM)
Noon (12:00 - 5:59 PM)
Morning (10:00 - 11:59 AM)
5
Night (6:00 - 9:59 PM)
Noon (12:00 - 5:59 PM)
Morning (10:00 - 11:59 AM)
4
Night (6:00 - 9:59 PM)
Noon (12:00 - 5:59 PM)
Morning (10:00 - 11:59 AM)
3
Night (6:00 - 9:59 PM)
Noon (12:00 - 5:59 PM)
Morning (10:00 - 11:59 AM)
2
Night (6:00 - 9:59 PM)
Noon (12:00 - 5:59 PM)
Morning (10:00 - 11:59 AM)
1
Night (6:00 - 9:59 PM)
Noon (12:00 - 5:59 PM)
Morning (10:00 - 11:59 AM)
2.95
3.44
3.42
2.98
3.39
3.01
3
3.31
2.85
3.13
3.16
2.74
2.95
3.21
3.38
3.03
3.26
3.41
2.89
3.04
3.13
0
0.5
1
1.5
2
2.5
3
3.5
4
Count



The query results analyze peak order and sales trends for each day of the week in 2023, highlighting that the **noon period (12:00 PM - 5:59 PM) consistently shows the highest average orders and sales**, with **Sundays reaching a peak** of 21 orders and $3,034.79 in sales. This analysis **helps determine the optimal times for limited-time deals or flash sales** to maximize their impact. By identifying that the **noon period generates the highest average sales**, businesses can strategically schedule promotions, such as a 20% discount from 12:00 PM to 2:00 PM, to

capitalize on increased customer traffic. This ensures that promotions align with peak hours, **boosting the chances of immediate purchases** and **driving higher sales and revenue.**

3.5 Tan Wan Yin

3.5.1 Monthly Analysis of Category Orders with Year-Over-Year (2021-2023) Comparisons

```
SPOOL "C:\Users\Wan Yin\Documents\query\query1.txt";
SET LINESIZE 132
SET PAGESIZE 50

-- Prompt user for the month input
ACCEPT v_month PROMPT 'Enter the month (01-12) for the sales details
with category: '

-- Define title for the output
TTITLE CENTER
'-------------------------------------------------------------------
----------------------------------' SKIP 1 -
       CENTER '     MONTHLY ANALYSIS OF CATEGORY ORDERS BY  ' SKIP 1
-
       CENTER '     YEAR-OVER-YEAR (2021-2023) COMPARISON IN MONTH:
&v_month ' SKIP 1 -
       CENTER
'-------------------------------------------------------------------
----------------------------------' SKIP 1 -
       LEFT   'DATE: ' _DATE -
       RIGHT  'PAGE: ' FORMAT 999 SQL.PNO SKIP 2

-- Format column headers and data
COLUMN Category HEADING 'CATEGORY' FORMAT A15
COLUMN Orders_2021 HEADING '2021-&v_month|ORDERS' FORMAT 99999
COLUMN Orders_2022 HEADING '2022-&v_month|ORDERS' FORMAT 99999
COLUMN Orders_2023 HEADING '2023-&v_month|ORDERS' FORMAT 99999
COLUMN "Avg Quantity" HEADING 'AVERAGE|ORDERS' FORMAT 99999
COLUMN "YoY Growth 21-22 (%)" HEADING 'YoY GROWTH|21-22 (%)' FORMAT
999.99
COLUMN "YoY Growth 22-23 (%)" HEADING 'YoY GROWTH|22-23 (%)' FORMAT
999.99
COLUMN "Average Growth Rate (%)" HEADING 'AVG GROWTH|RATE (%)' FORMAT
999.99
COLUMN "Projected Orders 2024" HEADING 'PROJECTED |ORDERS 2024'
FORMAT 99999
COLUMN "Projected Growth 2024 (%)" HEADING 'PROJECTED  |GROWTH
2024(%)' FORMAT 999.99
```

```
-- Create or replace views for each year with total number of orders
and percentage of total sales
CREATE OR REPLACE VIEW Ranked_Sales_2021 AS
SELECT
    m.categoryName AS Category,
    SUM(sf.quantity) AS TotalOrders_2021
FROM SalesFact sf
JOIN Date_dim d ON sf.DATE_KEY = d.date_key
JOIN menu_dim m ON sf.MENU_KEY = m.menu_key
WHERE d.cal_year = 2021
AND d.cal_month_year = '&v_month'
GROUP BY m.categoryName;

CREATE OR REPLACE VIEW Ranked_Sales_2022 AS
SELECT
    m.categoryName AS Category,
    SUM(sf.quantity) AS TotalOrders_2022
FROM SalesFact sf
JOIN Date_dim d ON sf.DATE_KEY = d.date_key
JOIN menu_dim m ON sf.MENU_KEY = m.menu_key
WHERE d.cal_year = 2022
AND d.cal_month_year = '&v_month'
GROUP BY m.categoryName;

CREATE OR REPLACE VIEW Ranked_Sales_2023 AS
SELECT
    m.categoryName AS Category,
    SUM(sf.quantity) AS TotalOrders_2023
FROM SalesFact sf
JOIN Date_dim d ON sf.DATE_KEY = d.date_key
JOIN menu_dim m ON sf.MENU_KEY = m.menu_key
WHERE d.cal_year = 2023
AND d.cal_month_year = '&v_month'
GROUP BY m.categoryName;

-- Combine results from different years with rankings
CREATE OR REPLACE VIEW Combined_Sales AS
SELECT
    COALESCE(s2021.Category, s2022.Category, s2023.Category) AS
Category,
    COALESCE(s2021.TotalOrders_2021, 0) AS Orders_2021,
    COALESCE(s2022.TotalOrders_2022, 0) AS Orders_2022,
    COALESCE(s2023.TotalOrders_2023, 0) AS Orders_2023,
```

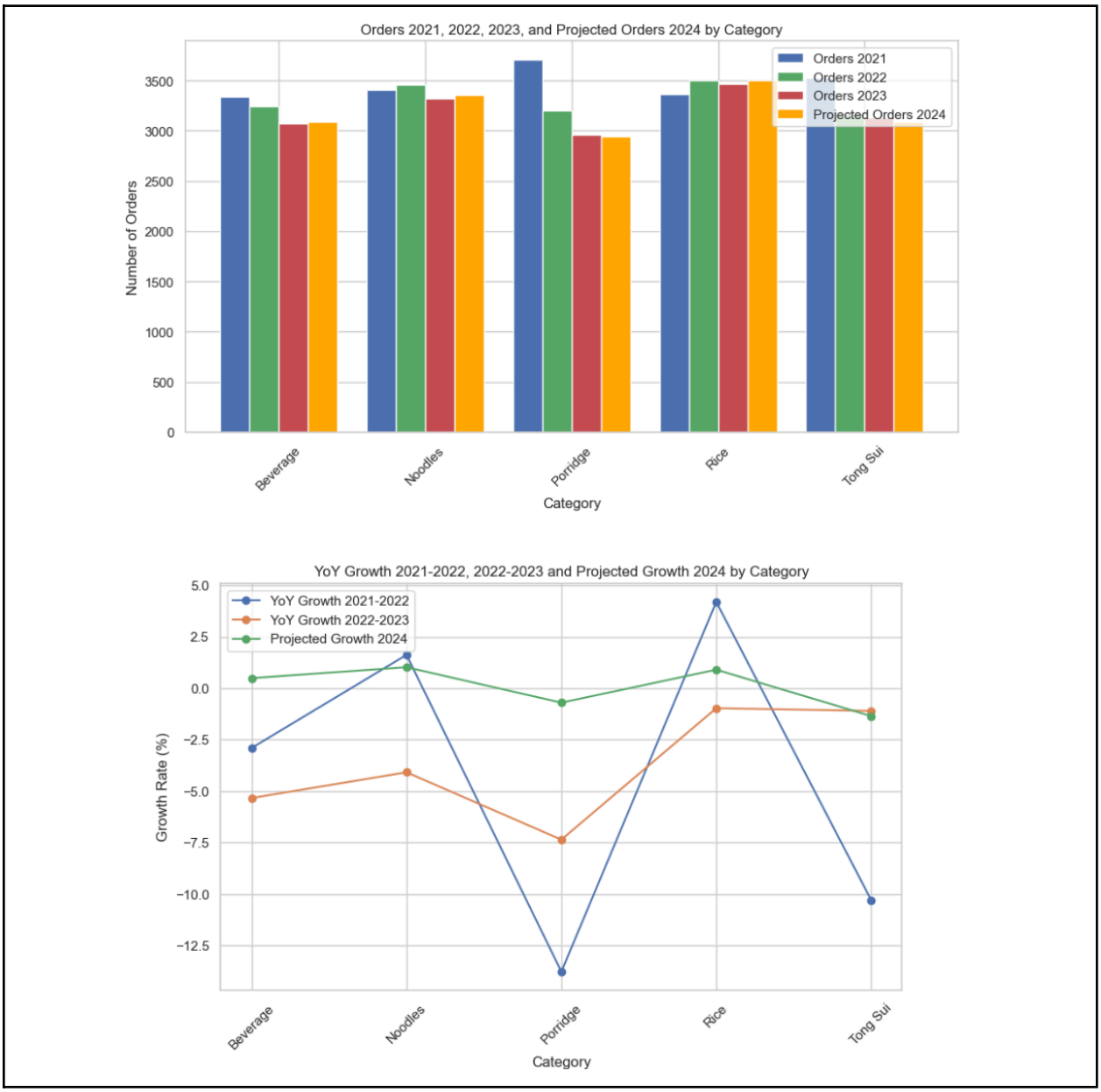
```
    ROUND((COALESCE(s2021.TotalOrders_2021, 0) +
COALESCE(s2022.TotalOrders_2022, 0) +
COALESCE(s2023.TotalOrders_2023, 0)) / 3, 2) AS "Avg Quantity",
    ROUND(((COALESCE(s2022.TotalOrders_2022, 0) -
COALESCE(s2021.TotalOrders_2021, 0)) / NULLIF(s2021.TotalOrders_2021,
0)) * 100, 2) AS "YoY Growth 21-22 (%)",
    ROUND(((COALESCE(s2023.TotalOrders_2023, 0) -
COALESCE(s2022.TotalOrders_2022, 0)) / NULLIF(s2022.TotalOrders_2022,
0)) * 100, 2) AS "YoY Growth 22-23 (%)",
    ROUND((ROUND(((COALESCE(s2022.TotalOrders_2022, 0) -
COALESCE(s2021.TotalOrders_2021, 0)) / NULLIF(s2021.TotalOrders_2021,
0)) * 100, 2) +
          ROUND(((COALESCE(s2023.TotalOrders_2023, 0) -
COALESCE(s2022.TotalOrders_2022, 0)) / NULLIF(s2022.TotalOrders_2022,
0)) * 100, 2)) / 2, 2) AS "Average Growth Rate (%)"
FROM Ranked_Sales_2021 s2021
FULL OUTER JOIN Ranked_Sales_2022 s2022 ON s2021.Category =
s2022.Category
FULL OUTER JOIN Ranked_Sales_2023 s2023 ON COALESCE(s2021.Category,
s2022.Category) = s2023.Category;

-- Final output query with additional metrics including projected YoY
Growth for 2023-2024
SELECT
    Category,
    Orders_2021,
    Orders_2022,
    Orders_2023,
    "Avg Quantity",
    "YoY Growth 21-22 (%)",
    "YoY Growth 22-23 (%)",
    "Average Growth Rate (%)",
    ROUND("Avg Quantity" * (1 + "Average Growth Rate (%)" / 100), 2)
AS "Projected Orders 2024",
    ROUND((ROUND("Avg Quantity" * (1 + "Average Growth Rate (%)" /
100), 2) - Orders_2023) / NULLIF(Orders_2023, 0) * 100, 2) AS
"Projected Growth 2024 (%)"
FROM Combined_Sales
ORDER BY Category;

SPOOL OFF;
```

```
Enter the month (01-12) for the sales details with category: 02
old   9: AND d.cal_month_year = '&v_month'
new   9: AND d.cal_month_year = '02'

View created.

old   9: AND d.cal_month_year = '&v_month'
new   9: AND d.cal_month_year = '02'

View created.

old   9: AND d.cal_month_year = '&v_month'
new   9: AND d.cal_month_year = '02'

View created.


View created.
```

MONTHLY ANALYSIS OF CATEGORY ORDERS BY
YEAR-OVER-YEAR (2021-2023) COMPARISON IN MONTH: 02

DATE: 21-SEP-24 PAGE: 1

CATEGORY	2021-02 ORDERS	2022-02 ORDERS	2023-02 ORDERS	AVERAGE ORDERS	YoY GROWTH 21-22 (%)	YoY GROWTH 22-23 (%)	AVG GROWTH RATE (%)	PROJECTED ORDERS 2024	PROJECTED GROWTH 2024(%)
BEVERAGE	3342	3245	3072	3220	-2.90	-5.33	-4.12	3087	.49
NOODLES	3404	3459	3318	3394	1.62	-4.08	-1.23	3352	1.02
PORRIDGE	3707	3197	2962	3289	-13.76	-7.35	-10.56	2941	-.70
RICE	3364	3505	3471	3447	4.19	-.97	1.61	3502	.90
TONG SUI	3531	3168	3133	3277	-10.28	-1.10	-5.69	3091	-1.35


Orders 2021, 2022, 2023, and Projected Orders 2024 by Category
Orders 2021
Orders 2022
Orders 2023
Projected Orders 2024
Number of Orders
0
500
1000
1500
2000
2500
3000
3500
Beverage
Noodles
Porridge
Rice
Tong Sui
Category
YoY Growth 2021-2022, 2022-2023 and Projected Growth 2024 by Category
YoY Growth 2021-2022
YoY Growth 2022-2023
Projected Growth 2024
Growth Rate (%)
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
Beverage
Noodles
Porridge
Rice
Tong Sui
Category

In the report, the Beverage category declined from 3,342 orders in 2021 to 3,072 in 2023, with a YoY decrease of 5.33% from 2022. This highlights the need for targeted marketing strategies, such as **seasonal promotions and limited-time offers** to boost sales. The Noodles category showed slight growth, peaking at 3,459 orders in 2022 with a YoY growth of 1.62%. However, a drop in 2023 suggests market saturation like **introducing new flavors like black pepper** could enhance customer interest.The Porridge category experienced a significant drop of 13.76% from 2021 to 2022, indicating a need for **product innovation**. Rice grew from 3,364 orders in 2021 to 3,471 in 2023, reflecting a 4.19% YoY growth from 2021 to 2022, although it saw a slight decline of 0.97% afterward. **Bundle packages** with complementary products like beverage discount RM1 of beverage in each order could further balance the overall category sales. Lastly, Tong Sui faced a decline of 10.28% YoY, suggesting a need for **revitalization strategies** to attract customers back.

Looking ahead to 2024, projected orders for Rice, Beverage, and Tong Sui suggest potential growth with 3,502, 3,087, and 3,091 orders, respectively. This positive outlook for 2024 emphasizes the necessity for targeted strategies, especially for Beverage and Tong Sui to foster recovery and drive higher sales. Overall,these categories can capitalize on projected growth opportunities in the upcoming year.

3.5.2 Weekend vs Weekday Customer Distribution of Top 20% Menu Item Order in Quarter across year (2021-2023)

```
SPOOL "C:\Users\Wan Yin\Documents\query\qquery2.txt";

SET LINESIZE 132
SET PAGESIZE 40

ACCEPT v_quarter PROMPT 'Enter the quarter (Q1-Q4): '

TTITLE CENTER
'-------------------------------------------------------------------
---------------------------------' SKIP 1 -
       CENTER 'WEEKEND VS WEEKDAY CUSTOMER DISTRIBUTION ' SKIP 1 -
       CENTER 'OF TOP 20% MENU ITEM ORDER IN &v_quarter ACROSS YEAR
(2021-2023)' SKIP 1 -
       CENTER
'-------------------------------------------------------------------
---------------------------------' SKIP 1 -
       LEFT   'DATE: ' _DATE -
       RIGHT  'PAGE: ' FORMAT 999 SQL.PNO SKIP 2

-- Format columns
COLUMN Quarter HEADING 'QUARTER' FORMAT A10
COLUMN DayType HEADING 'DAY TYPE' FORMAT A15
COLUMN MenuItem HEADING 'MENU ITEM' FORMAT A30
COLUMN Age_Group HEADING 'AGE RANGE' FORMAT A15
COLUMN Total HEADING 'TOTAL ORDERS' FORMAT 99999

BREAK ON Quarter SKIP PAGE ON DayType

CREATE OR REPLACE VIEW AgeGroupSales AS (
    SELECT
        D.cal_year_quarter AS Quarter,
        CASE WHEN D.Weekday_Ind = 'N' THEN 'Weekend' ELSE 'Weekday'
END AS DayType,
        P.menuname AS MenuItem,
        SUM(SF.quantity) AS Total,
        CASE
            WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, C.dob) / 12) BETWEEN
11 AND 20 THEN '11-20'
            WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, C.dob) / 12) BETWEEN
21 AND 30 THEN '21-30'
            WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, C.dob) / 12) BETWEEN
31 AND 40 THEN '31-40'
```

```
            WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, C.dob) / 12) BETWEEN
41 AND 50 THEN '41-50'
            WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, C.dob) / 12) BETWEEN
51 AND 60 THEN '51-60'
            WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, C.dob) / 12) BETWEEN
61 AND 65 THEN '60-65'
            ELSE 'Over 65'
        END AS Age_Group
    FROM salesfact SF
    JOIN date_dim D ON SF.date_key = D.date_key
    JOIN customer_dim C ON SF.customer_key = C.customer_key
    JOIN menu_dim P ON SF.menu_key = P.menu_key
    WHERE D.cal_year_quarter IN ('2021-' || '&v_quarter', '2022-' ||
'&v_quarter', '2023-' || '&v_quarter')
    GROUP BY
        D.cal_year_quarter,
        D.weekday_ind,
        P.menuname,
        CASE
            WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, C.dob) / 12) BETWEEN
11 AND 20 THEN '11-20'
            WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, C.dob) / 12) BETWEEN
21 AND 30 THEN '21-30'
            WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, C.dob) / 12) BETWEEN
31 AND 40 THEN '31-40'
            WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, C.dob) / 12) BETWEEN
41 AND 50 THEN '41-50'
            WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, C.dob) / 12) BETWEEN
51 AND 60 THEN '51-60'
            WHEN TRUNC(MONTHS_BETWEEN(SYSDATE, C.dob) / 12) BETWEEN
61 AND 65 THEN '60-65'
            ELSE 'Over 65'
        END
);

CREATE OR REPLACE VIEW RankedMenuItems AS (
    SELECT
        Quarter,
        DayType,
        MenuItem,
        Age_Group,
        Total,
        ROW_NUMBER() OVER (
            PARTITION BY Quarter, DayType
            ORDER BY Total DESC
```

```
        ) AS rn,
        COUNT(DISTINCT MenuItem) OVER (PARTITION BY Quarter, DayType)
AS TotalMenuItems
    FROM AgeGroupSales
);

-- Create or replace the TopMenuItems view
CREATE OR REPLACE VIEW TopMenuItems AS (
    SELECT
        Quarter,
        DayType,
        MenuItem,
        Age_Group,
        Total,
        rn,
        CEIL(0.2 * TotalMenuItems) AS TopPercentCount
    FROM RankedMenuItems
    where rn<= CEIL(0.2 * TotalMenuItems)
);


SELECT
    Quarter,
    DayType,
    MenuItem,
    Age_Group,
    Total
FROM TopMenuItems
ORDER BY Quarter, DayType, Total DESC, MenuItem;

SET LINESIZE 132
SET PAGESIZE 40
TTITLE CENTER
'-------------------------------------------------------------------
---------------------------------' SKIP 1 -
       CENTER 'MENU ITEM LIST HAVE APPEARED MORE THAN ONE YEAR DURING
QUARTER: &v_quarter' SKIP 1 -
       CENTER
'-------------------------------------------------------------------
---------------------------------' SKIP 1 -
       LEFT   'DATE: ' _DATE -
       RIGHT  'PAGE: ' FORMAT 999 SQL.PNO SKIP 2

COLUMN MenuItem HEADING 'MENU ITEM' FORMAT A30
COLUMN FrequencyYear HEADING 'FREQUENCY|YEAR  ' FORMAT 99
```

```
COLUMN AverageWeekdayOrders HEADING 'AVERAGE  |WEEKDAYS ORDERS'
FORMAT 99999
COLUMN AverageWeekendOrders HEADING 'AVERAGE  |WEEKEND ORDERS' FORMAT
99999
COLUMN WeekendOrdersCount HEADING 'WEEKEND|ORDERS| COUNT' FORMAT 999
COLUMN WeekdaysOrdersCount HEADING 'WEEKDAYS|ORDERS| COUNT' FORMAT
999
COLUMN WeekendFocusAgeGroups HEADING 'WEEKEND|AGE GROUP' FORMAT A15
COLUMN WeekdayFocusAgeGroups HEADING 'WEEKDAYS|AGE GROUP' FORMAT A15

SELECT
    MenuItem,
    COUNT(DISTINCT SUBSTR(Quarter, 1, 4)) AS FrequencyYear,
    ROUND(SUM(CASE WHEN DayType = 'Weekend' THEN Total ELSE 0 END) /
NULLIF(SUM(CASE WHEN DayType = 'Weekend' THEN 1 ELSE 0 END), 0)) AS
AverageWeekendOrders,
    ROUND(SUM(CASE WHEN DayType = 'Weekday' THEN Total ELSE 0 END) /
NULLIF(SUM(CASE WHEN DayType = 'Weekday' THEN 1 ELSE 0 END), 0)) AS
AverageWeekdayOrders,
    SUM(CASE WHEN DayType = 'Weekend' THEN 1 ELSE 0 END) AS
WeekendOrdersCount,
    SUM(CASE WHEN DayType = 'Weekday' THEN 1 ELSE 0 END) AS
WeekdaysOrdersCount,
    LISTAGG(CASE WHEN DayType = 'Weekend' THEN Age_Group END, ', ')
WITHIN GROUP (ORDER BY Age_Group) AS WeekendFocusAgeGroups,
    LISTAGG(CASE WHEN DayType = 'Weekday' THEN Age_Group END, ', ')
WITHIN GROUP (ORDER BY Age_Group) AS WeekdayFocusAgeGroups
FROM TopMenuItems
GROUP BY MenuItem
HAVING COUNT(DISTINCT SUBSTR(Quarter, 1, 4)) > 1
ORDER BY FrequencyYear DESC;

CLEAR COLUMN;
CLEAR BREAK;
TTITLE OFF;
SPOOL OFF;
```

```
Enter the quarter (Q1-Q4): Q2
old  20:     WHERE D.cal_year_quarter IN ('2021-' || '&v_quarter', '2022-' || '&v_quarter', '2023-' ||
'&v_quarter')
new  20:     WHERE D.cal_year_quarter IN ('2021-' || 'Q2', '2022-' || 'Q2', '2023-' || 'Q2')

View created.
View created.
View created.
```

WEEKEND VS WEEKDAY CUSTOMER DISTRIBUTION
OF TOP 20% MENU ITEM ORDER IN Q2 ACROSS YEAR (2021-2023)

DATE: 21-SEP-24
PAGE: 1

QUARTER	DAY TYPE	MENU ITEM	AGE RANGE	TOTAL ORDERS
2021-Q2	Weekday	ALMOND SOUP	41-50	481
		SEAFOOD CLAYPOT RICE	51-60	417
		PINEAPPLE FRIED RICE	41-50	407
		LUO HAN GUO SOUP	41-50	402
		CHICKEN PORRIDGE	51-60	390
		HOKKIEN MEE	41-50	389
		DURIAN SOUP	41-50	386
		YEE MEE	41-50	386
	Weekend	YANGZHOU FRIED RICE	51-60	229
		PINEAPPLE FRIED RICE	51-60	219
		HOT CHOCOLATE	51-60	174
		PAN MEE	51-60	174
		CLAYPOT RICE	51-60	168
		SEAFOOD CLAYPOT RICE	51-60	168
		HAINANESE CHICKEN RICE	51-60	165
		LAKSA	51-60	165

WEEKEND VS WEEKDAY CUSTOMER DISTRIBUTION
OF TOP 20% MENU ITEM ORDER IN Q2 ACROSS YEAR (2021-2023)

DATE: 21-SEP-24
PAGE: 2

QUARTER	DAY TYPE	MENU ITEM	AGE RANGE	TOTAL ORDERS
2022-Q2	Weekday	BLACK SESAME SOUP	51-60	473
		FRUIT PUNCH	41-50	466
		MUSHROOM PORRIDGE	51-60	438
		SEAFOOD CLAYPOT RICE	51-60	437
		YEE MEE	51-60	430
		TOM YUM FRIED RICE	51-60	420
		CURRY MEE	51-60	415
		HOKKIEN MEE	41-50	415
	Weekend	WAN TAN MEE	51-60	227
		CLAYPOT RICE	51-60	209
		PORK BONE PORRIDGE	51-60	208
		SWEET CORN PORRIDGE	51-60	191
		YANGZHOU FRIED RICE	41-50	188
		TOM YUM FRIED RICE	41-50	185
		HOT CHOCOLATE	51-60	184
		CHINESE TEA	51-60	183

WEEKEND VS WEEKDAY CUSTOMER DISTRIBUTION
OF TOP 20% MENU ITEM ORDER IN Q2 ACROSS YEAR (2021-2023)

DATE: 21-SEP-24
PAGE: 3

QUARTER	DAY TYPE	MENU ITEM	AGE RANGE	TOTAL ORDERS
2023-Q2	Weekday	HOT CHOCOLATE	51-60	382
		CLAYPOT RICE	51-60	379
		YANGZHOU FRIED RICE	51-60	373
		CHINESE TEA	41-50	367
		TOM YUM FRIED RICE	41-50	365
		MUSHROOM PORRIDGE	51-60	364
		PINEAPPLE FRIED RICE	51-60	363
		FRIED RICE WITH SALTED FISH	51-60	358
	Weekend	YEE MEE	51-60	218
		MANGO SAGO	51-60	213
		HOKKIEN MEE	51-60	206
		CHAR KWAY TEOW	41-50	204
		FISH HEAD BEE HOON	41-50	188
		WAN TAN MEE	41-50	180
		HAINANESE CHICKEN RICE	41-50	178
		SAGO PUDDING	51-60	177

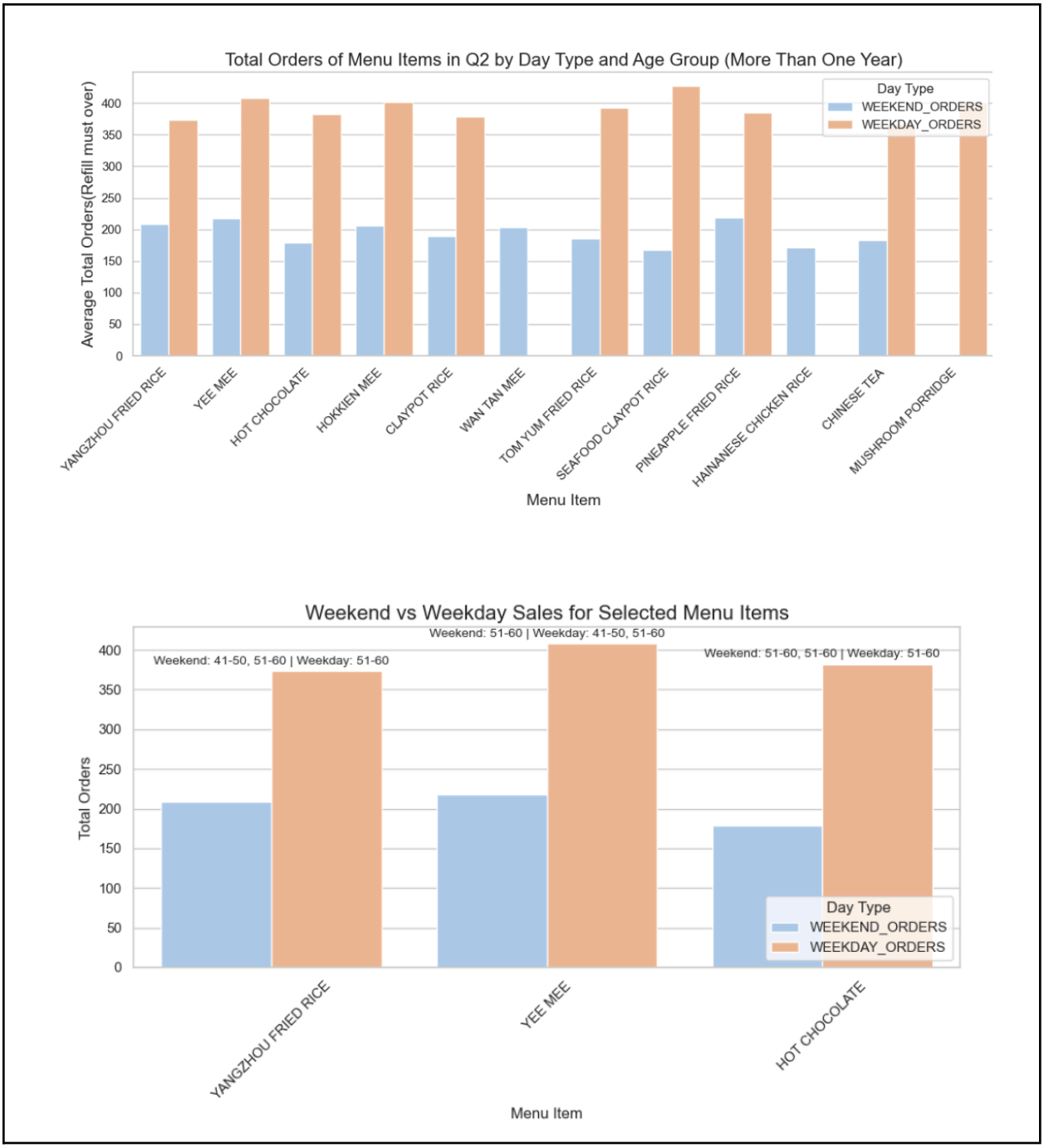
48 rows selected.

MENU ITEM LIST HAVE APPEARED MORE THAN ONE YEAR DURING QUARTER: Q2

DATE: 21-SEP-24
PAGE: 1

MENU ITEM	FREQUENCY YEAR	AVERAGE WEEKEND ORDERS	AVERAGE WEEKDAYS ORDERS	WEEKEND ORDERS COUNT	WEEKDAYS ORDERS COUNT	WEEKEND AGE GROUP	WEEKDAYS AGE GROUP
YANGZHOU FRIED RICE	3	209	373	2	1	41-50, 51-60	51-60
YEE MEE	3	218	408	1	2	51-60	41-50, 51-60
HOT CHOCOLATE	3	179	382	2	1	51-60, 51-60	51-60
HOKKIEN MEE	3	206	402	1	2	51-60	41-50, 41-50
CLAYPOT RICE	3	189	379	2	1	51-60, 51-60	51-60
WAN TAN MEE	2	204		2	0	41-50, 51-60	
TOM YUM FRIED RICE	2	185	393	1	2	41-50	41-50, 51-60
SEAFOOD CLAYPOT RICE	2	168	427	1	2	51-60	51-60, 51-60
PINEAPPLE FRIED RICE	2	219	385	1	2	51-60	41-50, 51-60
HAINANESE CHICKEN RICE	2	172		2	0	41-50, 51-60	
CHINESE TEA	2	183	367	1	1	51-60	41-50
MUSHROOM PORRIDGE	2		401	0	2		51-60, 51-60

12 rows selected.



In the report, items like "YEE MEE" and "HOT CHOCOLATE" show higher total orders on weekdays, suggesting a preference for these options during lunch hours or midweek dining among certain age groups. Conversely, weekend favorites like "YANGZHOU FRIED RICE" and "CLAYPOT RICE" indicate a trend toward heartier meals during leisure time for the same demographic. To optimize inventory management for these items, particularly "YANGZHOU FRIED RICE" and "CLAYPOT RICE," businesses should implement advanced planning strategies, such as **increasing stock levels in**

advance for key ingredients like rice bags.This proactive approach can help prevent stockouts and **minimize waste** by ensuring fresh ingredients are available when needed.The restock level of these ingredients must exceed the volume of the ingredient that is used in average total orders.

Then, its specific look for the menu item like "YEE MEE" appeals to both weekdays and weekends, especially among the 51-60 age group, highlighting its role as a reliable dining choice. Conversely, "HOT CHOCOLATE" and "YANGZHOU FRIED RICE" demonstrate stronger weekday sales within the same demographic which indicates a preference for comforting options during busy lunch hours. These insights suggest that businesses could enhance their marketing strategies by promoting "YEE MEE" as a versatile dish while **targeting promotions** for "HOT CHOCOLATE" and "YANGZHOU FRIED RICE" specifically to the 51-60 age group on weekdays.

3.5.3 Comparative Analysis of Total Orders and Revenue By Menu Category

```
SPOOL "C:\Users\Wan Yin\Documents\query\query3.txt";
SET LINESIZE 132
SET PAGESIZE 40

ACCEPT year1 NUMBER PROMPT 'Enter the current year (2021-2023) for
comparison: '
ACCEPT year2 NUMBER PROMPT 'Enter the previous year (2020-2022) for
comparison: '

TTITLE CENTER
'-------------------------------------------------------------------
-----------------------------------' SKIP 1 -
       CENTER '   COMPARATIVE ANALYSIS OF TOTAL ORDERS AND REVENUE BY
MENU CATEGORY' SKIP 1 -
       CENTER '   DURING FESTIVAL BETWEEN '&year1' AND '&year2' '
SKIP 1 -
       CENTER
'-------------------------------------------------------------------
-----------------------------------' SKIP 1 -
       LEFT   'DATE: ' _DATE -
       RIGHT  'PAGE: ' FORMAT 999 SQL.PNO SKIP 2

COLUMN Festival HEADING 'FESTIVAL' FORMAT A10
COLUMN Category HEADING 'CATEGORY' FORMAT A8
COLUMN "&year1 Orders" HEADING '&year1|Orders' FORMAT 9999
COLUMN "&year2 Orders" HEADING '&year2|Orders' FORMAT 9999
COLUMN "&year1 Revenue" HEADING '&year1|Revenue' FORMAT 999,999.99
COLUMN "&year2 Revenue" HEADING '&year2|Revenue' FORMAT 999,999.99
COLUMN ProjectedAverageTotalRevenue HEADING
'Avg|Projected|Total|Revenue|(5%)' FORMAT 999,999.99
COLUMN ProjectedGrowth HEADING 'Projected|Growth|%' FORMAT 99999.99
COLUMN GrowthPercentage HEADING 'Growth|%' FORMAT 999.99
COLUMN GrowthRevenuePercentage HEADING 'Growth|Revenue|%' FORMAT
999.99
COLUMN AverageGrowthPercentage HEADING 'Avg|Growth|Orders|%' FORMAT
999.99
COLUMN AverageGrowthRevenuePercentage HEADING 'Avg|Growth|Revenue|%'
FORMAT 999.99
COLUMN AveProjectedGrowth HEADING 'Avg|Projected|Growth|%' FORMAT
999.99

BREAK ON Festival SKIP 1
COMPUTE SUM OF "Current Year-&year1 Orders" ON Festival
```

```
COMPUTE SUM OF "Previous Year-&year2 Orders" ON Festival
COMPUTE SUM OF "Current Year-&year1 Revenue" ON Festival
COMPUTE SUM OF "Previous Year-&year2 Revenue" ON Festival

CREATE OR REPLACE VIEW Festival_Year1_CategorySales AS
SELECT
    fm.simplified_festival_name AS festival_name,
    m.categoryName AS menu_category,
    SUM(s.Quantity) AS total_orders,
    SUM(s.lineTotal) AS total_revenue
FROM
    SALESFACT s
JOIN
    DATE_DIM d ON s.DATE_KEY = d.date_key
JOIN
    MENU_DIM m ON s.MENU_KEY = m.menu_key
JOIN
    PROMOTION_DIM p ON s.PROMO_KEY = p.promo_key
JOIN
    (
        SELECT DISTINCT
            p.promoName AS original_festival_name,
            CASE
                WHEN p.promoName LIKE '%NEW YEAR%' THEN 'CNY'
                WHEN p.promoName LIKE '%DEEPAVALI%' THEN 'DEEPAVALI'
                WHEN p.promoName LIKE '%HARI RAYA%' THEN 'HARI RAYA'
                ELSE NULL
            END AS simplified_festival_name
        FROM PROMOTION_DIM p
    ) fm ON p.promoName = fm.original_festival_name
WHERE
    fm.simplified_festival_name IS NOT NULL
    AND EXTRACT(YEAR FROM d.cal_date) = &year1
GROUP BY
    fm.simplified_festival_name,
    m.categoryName
ORDER BY
    festival_name, menu_category;

CREATE OR REPLACE VIEW Festival_Year2_CategorySales AS
SELECT
    fm.simplified_festival_name AS festival_name,
    m.categoryName AS menu_category,
    SUM(s.Quantity) AS total_orders,
    SUM(s.lineTotal) AS total_revenue
```

```
FROM
    SALESFACT s
JOIN
    DATE_DIM d ON s.DATE_KEY = d.date_key
JOIN
    MENU_DIM m ON s.MENU_KEY = m.menu_key
JOIN
    PROMOTION_DIM p ON s.PROMO_KEY = p.promo_key
JOIN
    (
        SELECT DISTINCT
            p.promoName AS original_festival_name,
            CASE
                WHEN p.promoName LIKE '%NEW YEAR%' THEN 'CNY'
                WHEN p.promoName LIKE '%DEEPAVALI%' THEN 'DEEPAVALI'
                WHEN p.promoName LIKE '%HARI RAYA%' THEN 'HARI RAYA'
                ELSE NULL
            END AS simplified_festival_name
        FROM PROMOTION_DIM p
    ) fm ON p.promoName = fm.original_festival_name
WHERE
    fm.simplified_festival_name IS NOT NULL
    AND EXTRACT(YEAR FROM d.cal_date) = &year2
GROUP BY
    fm.simplified_festival_name,
    m.categoryName
ORDER BY
    festival_name, menu_category;

CREATE OR REPLACE VIEW GrowthData AS
SELECT
    COALESCE(fyear1.festival_name, fyear2.festival_name) AS festival,
    COALESCE(fyear1.menu_category, fyear2.menu_category) AS category,
    COALESCE(fyear1.total_orders, 0) AS "&year1 Orders",
    COALESCE(fyear2.total_orders, 0) AS "&year2 Orders",
    ROUND((COALESCE(fyear1.total_orders, 0) -
COALESCE(fyear2.total_orders, 0)) /
NULLIF(COALESCE(fyear2.total_orders, 0), 0) * 100, 2) AS
GrowthPercentage,
    COALESCE(fyear1.total_revenue, 0) AS "&year1 Revenue",
    COALESCE(fyear2.total_revenue, 0) AS "&year2 Revenue",
    ROUND((COALESCE(fyear1.total_revenue, 0) -
COALESCE(fyear2.total_revenue, 0)) /
NULLIF(COALESCE(fyear2.total_revenue, 0), 0) * 100, 2) AS
GrowthRevenuePercentage,
```

```
    (COALESCE(fyear1.total_revenue, 0) +
COALESCE(fyear2.total_revenue, 0)) / 2 AS AverageTotalRevenue
FROM
    Festival_Year1_CategorySales fyear1
FULL OUTER JOIN
    Festival_Year2_CategorySales fyear2
ON
    fyear1.festival_name = fyear2.festival_name
    AND fyear1.menu_category = fyear2.menu_category;

SELECT
    festival,
    category,
    "&year1 Orders",
    "&year2 Orders",
    GrowthPercentage,
    CASE WHEN ROW_NUMBER() OVER (PARTITION BY festival ORDER BY
category) = 1 THEN
        ROUND(AVG(GrowthPercentage) OVER (PARTITION BY festival), 2)
    ELSE NULL END AS AverageGrowthPercentage,
    "&year1 Revenue",
    "&year2 Revenue",
    GrowthRevenuePercentage,
    CASE WHEN ROW_NUMBER() OVER (PARTITION BY festival ORDER BY
category) = 1 THEN
        ROUND(AVG(GrowthRevenuePercentage) OVER (PARTITION BY
festival), 2)
    ELSE NULL END AS AverageGrowthRevenuePercentage,
    AverageTotalRevenue * 1.05 AS ProjectedAverageTotalRevenue,
    ROUND(((AverageTotalRevenue * 1.05) - COALESCE("&year1 Revenue",
0)) / NULLIF(COALESCE("&year1 Revenue", 0), 0) * 100, 2) AS
ProjectedGrowth,
     CASE WHEN ROW_NUMBER() OVER (PARTITION BY festival ORDER BY
category) = 1 THEN
        ROUND(SUM(ROUND(((AverageTotalRevenue * 1.05) -
COALESCE("&year1 Revenue", 0)) / NULLIF(COALESCE("&year1 Revenue",
0), 0) * 100, 2)) OVER (PARTITION BY festival) / 5, 2)
    ELSE NULL END AS AveProjectedGrowth
FROM
    GrowthData
ORDER BY
    festival, category;
SPOOL OFF;
```

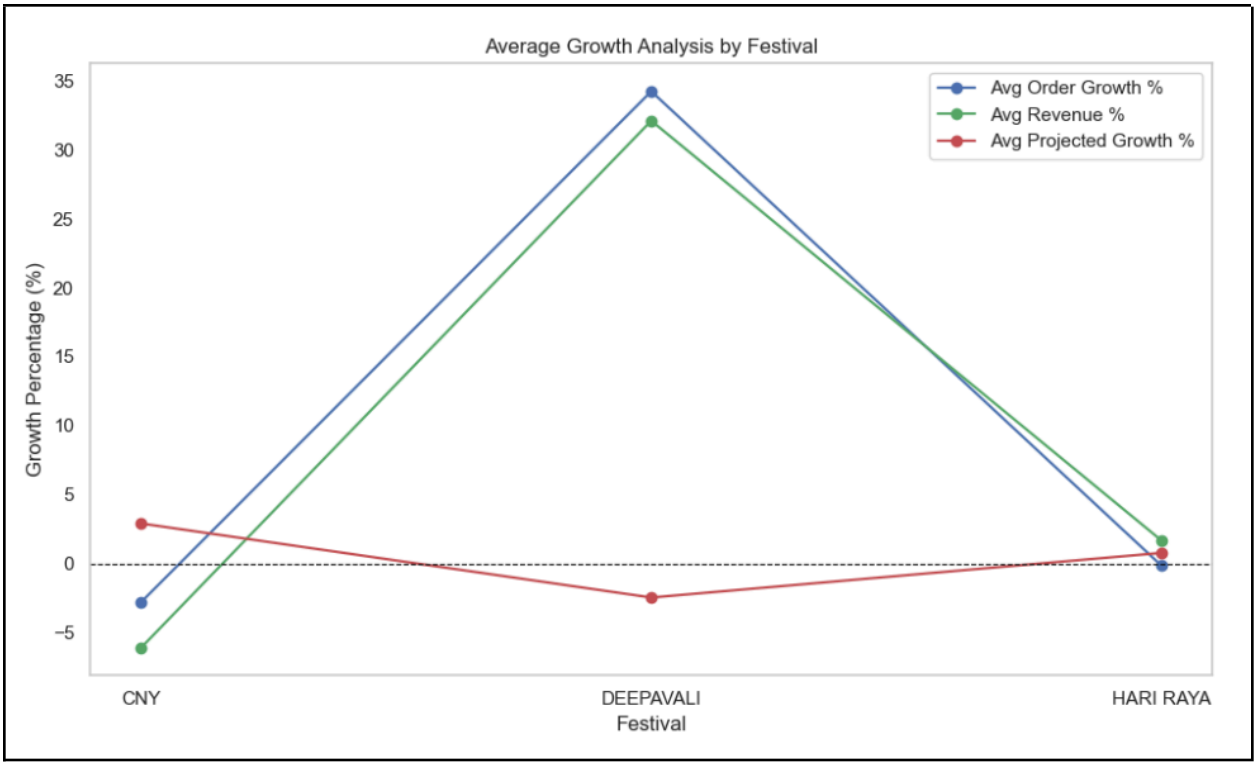
```
Enter the current year (2021-2023) for comparison: 2022
Enter the previous year (2020-2022) for comparison: 2021
old  29:     AND EXTRACT(YEAR FROM d.cal_date) = &year1
new  29:     AND EXTRACT(YEAR FROM d.cal_date) =       2022
View created.
old  29:     AND EXTRACT(YEAR FROM d.cal_date) = &year2
new  29:     AND EXTRACT(YEAR FROM d.cal_date) =       2021
View created.
old   5:     COALESCE(fyear1.total_orders, 0) AS "&year1 Orders",
new   5:     COALESCE(fyear1.total_orders, 0) AS "      2022 Orders",
old   6:     COALESCE(fyear2.total_orders, 0) AS "&year2 Orders",
new   6:     COALESCE(fyear2.total_orders, 0) AS "      2021 Orders",
old   8:     COALESCE(fyear1.total_revenue, 0) AS "&year1 Revenue",
new   8:     COALESCE(fyear1.total_revenue, 0) AS "      2022 Revenue",
old   9:     COALESCE(fyear2.total_revenue, 0) AS "&year2 Revenue",
new   9:     COALESCE(fyear2.total_revenue, 0) AS "      2021 Revenue",
View created.

old   4:     "&year1 Orders",
new   4:     "      2022 Orders",
old   5:     "&year2 Orders",
new   5:     "      2021 Orders",
old  10:     "&year1 Revenue",
new  10:     "      2022 Revenue",
old  11:     "&year2 Revenue",
new  11:     "      2021 Revenue",
old  17:     ROUND(((AverageTotalRevenue * 1.05) - COALESCE("&year1 Revenue", 0)) / NULLIF(COALESCE("&year1 Revenue",
0), 0) * 100, 2) AS ProjectedGrowth,
new  17:     ROUND(((AverageTotalRevenue * 1.05) - COALESCE("      2022 Revenue", 0)) / NULLIF(COALESCE("      2022
Revenue", 0), 0) * 100, 2) AS ProjectedGrowth,
old  19:         ROUND(SUM(ROUND(((AverageTotalRevenue * 1.05) - COALESCE("&year1 Revenue", 0)) /
NULLIF(COALESCE("&year1 Revenue", 0), 0) * 100, 2)) OVER (PARTITION BY festival) / 5, 2)
new  19:         ROUND(SUM(ROUND(((AverageTotalRevenue * 1.05) - COALESCE("      2022 Revenue", 0)) / NULLIF(COALESCE("
2022 Revenue", 0), 0) * 100, 2)) OVER (PARTITION BY festival) / 5, 2)
```

COMPARATIVE ANALYSIS OF TOTAL ORDERS AND REVENUE BY MENU CATEGORY
DURING FESTIVAL BETWEEN 2022 AND 2021

DATE: 21-SEP-24 PAGE: 1

FESTIVAL	CATEGORY	2022 Orders	2021 Orders	Growth %	Avg Growth Orders %	2022 Revenue	2021 Revenue	Growth Revenue %	Avg Growth Revenue %	Projected Avg Total Revenue (+ 5%)	Projected Revenue Growth %	Projected Avg Revenue Growth %
CNY	BEVERAGE	2134	1990	7.24	-2.76	4,778.08	4,589.12	4.12	-6.06	4,917.78	2.92	8.58
	NOODLES	1783	1840	-3.10		17,275.20	18,307.95	-5.64		18,681.15	8.14	
	PORRIDGE	1947	2088	-6.75		13,882.40	15,894.02	-12.66		15,632.62	12.61	
	RICE	1956	2017	-3.02		20,966.40	22,518.45	-6.89		22,829.55	8.89	
	TONG SUI	1904	2073	-8.15		12,264.24	13,510.43	-9.22		13,531.70	10.33	
DEEPAVALI	BEVERAGE	1795	1608	11.63	34.21	3,727.27	3,199.33	16.50	32.08	3,636.47	-2.44	-7.26
	NOODLES	2125	1588	33.82		20,673.75	15,936.75	29.72		19,220.51	-7.03	
	PORRIDGE	2046	1663	23.03		14,055.52	11,864.73	18.46		13,608.13	-3.18	
	RICE	2138	1522	40.47		21,285.00	15,465.75	37.63		19,294.14	-9.35	
	TONG SUI	2193	1353	62.08		13,360.64	8,452.55	58.07		11,451.92	-14.29	
HARI RAYA	BEVERAGE	1857	1704	8.98	-.12	3,932.92	3,617.40	8.72	1.70	3,963.92	.79	4.28
	NOODLES	1829	1936	-5.53		18,106.50	18,392.25	-1.55		19,161.84	5.83	
	PORRIDGE	1802	1850	-2.59		12,419.70	12,288.84	1.06		12,971.98	4.45	
	RICE	1722	1837	-6.26		17,539.50	18,720.75	-6.31		19,036.63	8.54	
	TONG SUI	1905	1818	4.79		11,701.19	10,981.24	6.56		11,908.28	1.77	

15 rows selected.



In this report, the analysis of the festival periods in 2021 and 2022 reveals a positive correlation between average order growth and average revenue growth, indicating a balanced relationship between the number of orders and revenue. Notably, during Deepavali, there was substantial growth in both average orders (34.21%) and average revenue (32.08%), suggesting that pricing strategies were effectively aligned with customer demand. This alignment indicates that the pricing adjustments made during this festival successfully drove both sales volume and revenue.

Looking ahead to projected growth in 2024, increasing overall menu prices by 5% could result in a significant drop in revenue, despite a potential increase in the number of total orders. This underscores the need for the company to **reconsider its pricing strategy** like increasing slightly to 2 or 3 % to ensure sustainable growth while maintaining profitability.