TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

Assignment Title

Subject: BAIT 3003 Data Warehouse Technology

2024/2025

Student's name/ ID Number	Signature
Chong Xu Ming / 22WMR14305	chong
Ten Wei Kang / 22WMR14227	kang
Tong Chun Mun/22WMR14450	tong
Terence Tiu Chuan Jie/22WMR14462	terence

BAIT3003 Data Warehouse Technology

Assignment Assessment Form

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

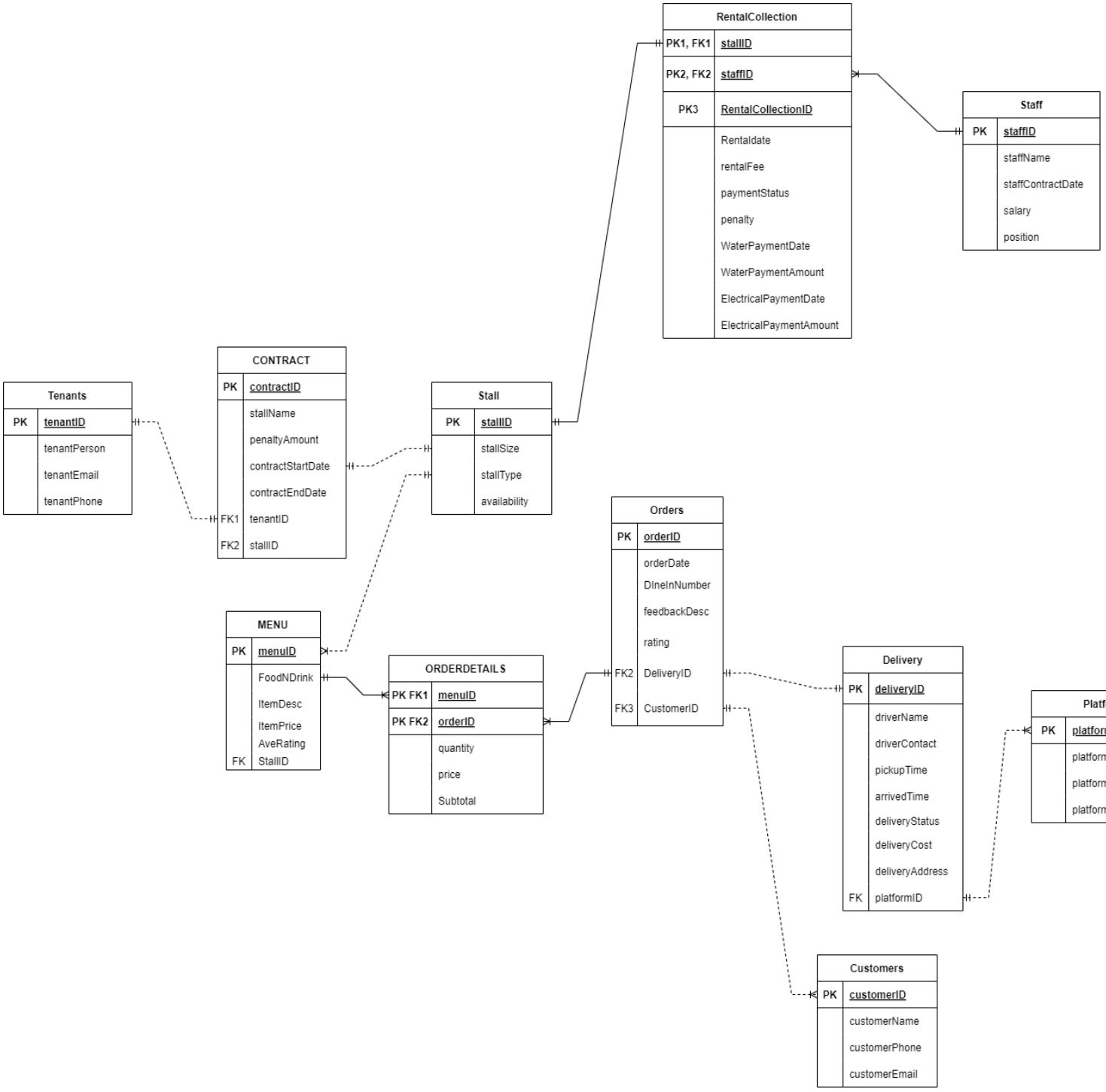
Group Member	**Task 3 marks**	**Total marks**
1. Chong Xu Ming	**( )**	**( )**
2. Ten Wei Kang	**( )**	**( )**
3. Tong Chun Mun	**( )**	**( )**
4.Terence Tiu Chuan Jie	**( )**	**( )**

Table of Contents

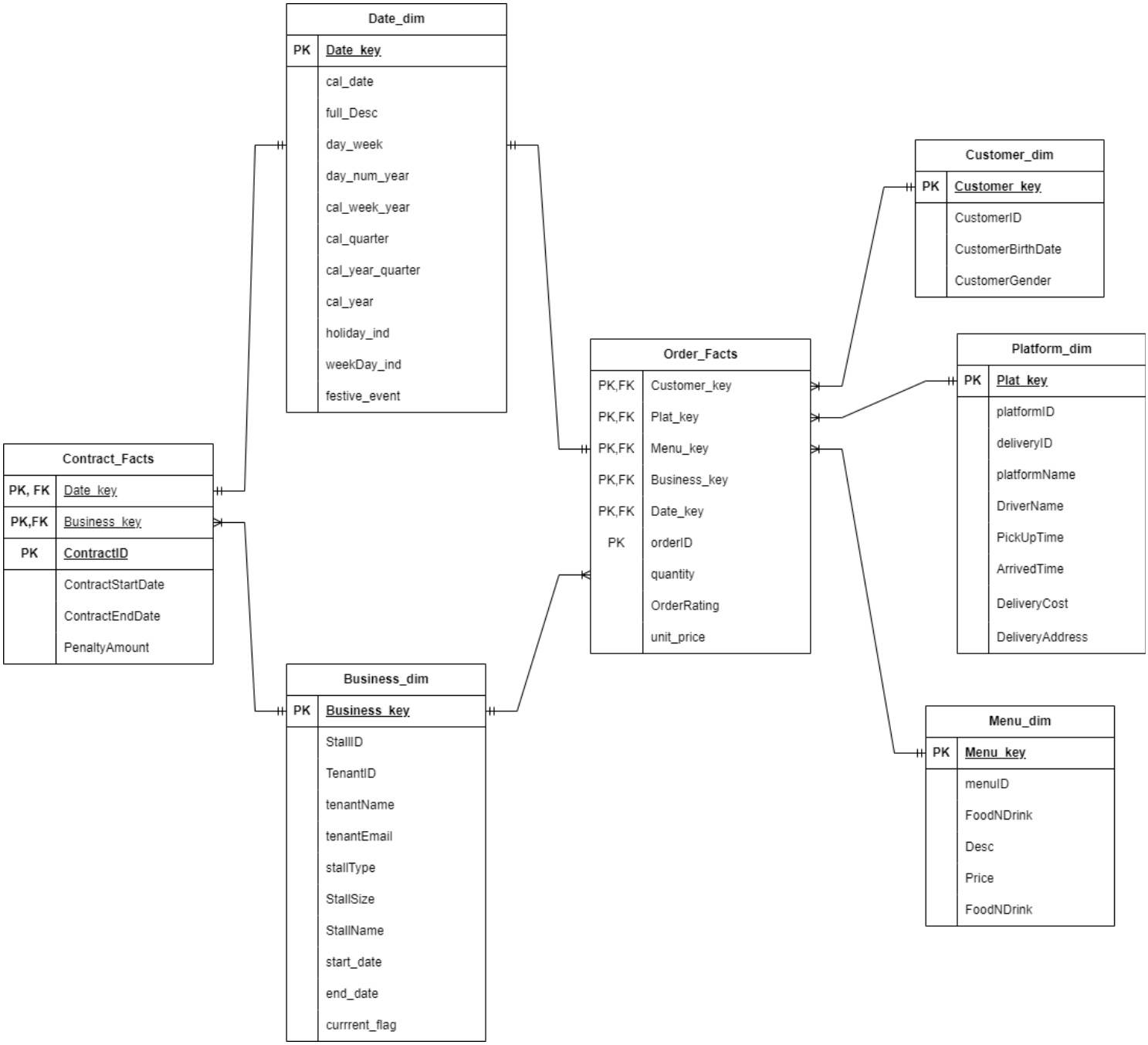
Chapter 1 Design of Data Warehouse

1.1 Logical Design

1.1.1 Original Database (Entity Relationship Diagram)



1.1.2 Star Schema Dimension and Fact Tables



1.2 Physical Design

1.2.1 Dimension Tables

Date Dimension Table

```
drop table Date_dim;
CREATE TABLE Date_dim
(
    date_key            NUMBER NOT NULL,    -- Primary Key, Running Number
    cal_date            DATE NOT NULL,      -- Calendar Date
    full_desc           VARCHAR2(31),       -- Full spelling of the date
    day_num_year        NUMBER(3),          -- 1 to 366 (Day number of the year)
    cal_week_year       NUMBER(2),          -- 1 to 52 (Week of the year)
    cal_quarter         CHAR(2),            -- Q1 to Q4 (Quarter of the year)
    cal_year_quarter    CHAR(7),            -- 'YYYY-QX' (Year and Quarter)
    cal_year            NUMBER(4),          -- Calendar Year
    holiday_ind         CHAR(1),            -- 'Y'/'N' (Holiday indicator)
    festive_event       VARCHAR2(28),       -- Name of festive event (if any)
    weekDay_ind         CHAR(1),            -- 'Y'/'N' (Is a weekday)

    CONSTRAINT PK_date_key PRIMARY KEY(date_key)
);
```

Business Dimension Table

```
create table business_dim
(business_key number NOT NULL,
 stallID   varchar(4) NOT NULL,
 tenantID  varchar(4) NOT NULL,
 tenantName varchar(20) NOT NULL,
 tenantEmail varchar(35) NOT NULL,
 stallSize varchar(35) NOT NULL,
 stallType varchar(35) NOT NULL,
 stallName varchar(35) NOT NULL,
constraint PK_business_key primary key(business_key)
);
```

Customer Dimension Table

```
CREATE TABLE Customer_dim (
  CUSTOMER_KEY     NUMBER NOT NULL,
  CUSTOMER_ID      NUMBER NOT NULL,
  CUSTOMER_BIRTHDATE DATE  NOT NULL,
  CUSTOMER_GENDER  CHAR(1)  NOT NULL,
  CONSTRAINT PK_customer PRIMARY KEY (CUSTOMER_KEY)
```

```
);
```

Platform Dimension Table

```
CREATE TABLE Platform_dim (
    platform_key NUMBER  not null,
    platformID varchar(5) NOT NULL,
    deliveryID NUMBER not null,
    PlatformName varchar(10) NOT NULL,
    DriverName varchar(20) not null,
    PickUpTime varchar(10) not null,
    ArrivedTime varchar(10) not null,
    DeliveryCost number(4,2) not null,
    DeliveryAddress varchar(50) not null,
CONSTRAINT PK_Platform_dim PRIMARY KEY(platform_key)
);
```

Menu Dimension Table

```
CREATE TABLE MENU_DIM (
  MENU_KEY     NUMBER NOT NULL ,
  menuID       varchar(4) NOT NULL,
  FOODNDRINK   VARCHAR(25) NOT NULL,
  ITEM_DESC    VARCHAR(100) NOT NULL,
  ITEM_PRICE   NUMBER(5,2) NOT NULL,
  CONSTRAINT PK_MENU PRIMARY KEY (MENU_KEY)
);
```

1.2.2 Fact Tables

Contract Fact Table

```
create table contract_fact
(date_key          number      not null,
 business_key      number      not null,
 contractID        varchar2(5) not null,
 contractStartDate date        not null,
 contractEndDate   date        not null,
 penaltyAmount     number(6,2) not null,
constraint PK_contract_fact primary key(date_key,business_key,contractID),
constraint FK_contract_date_key foreign key(date_key) references date_dim(date_key),
constraint FK_contract_business_key foreign key(business_key) references
business_dim(business_key)
);
```

Order Fact Table

```
CREATE TABLE Order_Facts (
    CUSTOMER_KEY NUMBER NOT NULL,
    Platform_key NUMBER NOT NULL,
    Menu_key NUMBER NOT NULL,
    Business_key NUMBER NOT NULL,
    DATE_KEY NUMBER NOT NULL,
    orderID NUMBER NOT NULL,
    quantity NUMBER NOT NULL,
    OrderRating NUMBER NOT NULL,
    unit_price NUMBER(5, 2) NOT NULL,
    CONSTRAINT PK_ORDER_fact PRIMARY KEY (date_key, customer_key, platform_key, orderid,
Menu_key, business_key),
    CONSTRAINT FK_date_key FOREIGN KEY (date_key) REFERENCES date_dim (date_key),
    CONSTRAINT FK_customer_key FOREIGN KEY (customer_key) REFERENCES customer_dim (customer_key),
    CONSTRAINT FK_Plat_key FOREIGN KEY (Platform_key) REFERENCES platform_dim (Platform_key),
    CONSTRAINT FK_Business_key FOREIGN KEY (Business_key) REFERENCES Business_dim (Business_key),
    CONSTRAINT FK_Menu_key FOREIGN KEY (Menu_key) REFERENCES Menu_dim (Menu_key)
);
```

Chapter 2 Extract, Transform, Load Process

2.1 Script for initial loading

Date Dimension Table

```
Drop sequence Date_seq;
Create sequence Date_seq
START WITH 100001
INCREMENT BY 1;


-- generate data
DECLARE
    startDate           DATE := TO_DATE('01/01/2014', 'dd/mm/yyyy'); -- Start date: January 1,
2014
    endDate             DATE := TO_DATE('31/12/2023', 'dd/mm/yyyy'); -- End date: December 31,
2023
    v_CAL_DATE          DATE;
    v_FULL_DESC         VARCHAR2(40);
    v_DAY_NUM_YEAR      NUMBER(3);
    v_CAL_WEEK_YEAR     NUMBER(2);
    v_CAL_QUARTER       CHAR(2);
    v_CAL_YEAR_QUARTER  CHAR(7);
    v_CAL_YEAR          NUMBER(4);
    v_HOLIDAY_IND       CHAR(1);
    v_WEEKDAY_IND       CHAR(1);
    v_FESTIVE_EVENT     VARCHAR2(25);
    counter             NUMBER := 0;

BEGIN
    WHILE (startDate <= endDate) LOOP
        -- Assign values to the variables based on startDate
        v_CAL_DATE := startDate;
        v_FULL_DESC := SUBSTR(TO_CHAR(startDate, 'Year') || ' ' || TO_CHAR(startDate, 'MONTH') ||
' ' || TO_CHAR(startDate, 'DD'), 1, 31);

        v_DAY_NUM_YEAR := TO_CHAR(startDate, 'DDD');  -- Day number of the year
        v_CAL_WEEK_YEAR := TO_CHAR(startDate, 'IW');  -- Week number in the year
        v_CAL_QUARTER := TO_CHAR(startDate, 'Q');     -- Quarter of the year
        v_CAL_YEAR_QUARTER := TO_CHAR(startDate, 'YYYY-Q');  -- Year-Quarter format
        v_CAL_YEAR := TO_CHAR(startDate, 'YYYY');     -- Year

        -- Set holiday indicator to 'N' for now (can be updated for actual holidays)
        v_HOLIDAY_IND := 'N';

        -- Determine if the current day is a weekday or weekend
        IF TO_CHAR(startDate, 'D') BETWEEN 2 AND 6 THEN  -- Weekday (Mon-Fri)
```

```
            v_WEEKDAY_IND := 'Y';
        ELSE  -- Weekend (Sat-Sun)
            v_WEEKDAY_IND := 'N';
        END IF;

        -- Set festive event to NULL (can be updated with actual events)
        v_FESTIVE_EVENT := NULL;

        -- Insert the values into the Date_dim table
        INSERT INTO Date_dim (
            date_key,
            cal_date,
            full_desc,
            day_num_year,
            cal_week_year,
            cal_quarter,
            cal_year_quarter,
            cal_year,
            holiday_ind,
            weekDay_ind,
            festive_event
        )
        VALUES (
            date_seq.NEXTVAL,
            v_CAL_DATE,
            v_FULL_DESC,
            v_DAY_NUM_YEAR,
            v_CAL_WEEK_YEAR,
            v_CAL_QUARTER,
            v_CAL_YEAR_QUARTER,
            v_CAL_YEAR,
            v_HOLIDAY_IND,
            v_WEEKDAY_IND,
            v_FESTIVE_EVENT
        );

        -- Increment the date by one day
        startDate := startDate + 1;
    END LOOP;
END;
/
EXEC Prod_Insert_Date_Dim;
```

Business Dimension Table

```
INSERT INTO business_dim
SELECT
```

```
    business_dim_seq.NEXTVAL,
    grouped_data.stallID,
    grouped_data.tenantID,
    grouped_data.tenantPerson,
    grouped_data.tenantEmail,
    grouped_data.stallSize,
    grouped_data.stallType,
    grouped_data.stallName
FROM (
    SELECT
        S.stallID,
        T.tenantID,
        MAX(T.tenantPerson) AS tenantPerson,
        MAX(T.tenantEmail) AS tenantEmail,
        MAX(S.stallSize) AS stallSize,
        MAX(S.stallType) AS stallType,
        MAX(C.stallName) AS stallName
    FROM new_stall S
    JOIN new_contract C ON S.stallID = C.stallID
    JOIN new_tenants T ON C.tenantID = T.tenantID
    GROUP BY S.stallID, T.tenantID
) grouped_data;

-- Add columns for start_date, end_date, and current_flag
ALTER TABLE business_dim
ADD (
    start_date DATE,                    -- Add without NOT NULL for now
    end_date DATE,                      -- End date for historical records
    current_flag CHAR(1) DEFAULT 'Y'    -- Current record flag, default is 'Y'
);

-- Update the start_date for each business record
UPDATE business_dim bd
SET bd.start_date = (
        SELECT MIN(nc.contractStartDate)
        FROM new_contract nc
        WHERE nc.stallID = bd.stallID
        AND nc.tenantID = bd.tenantID
    );


-- Update the start_date column to NOT NULL
ALTER TABLE business_dim
MODIFY start_date DATE NOT NULL;

-- Update the end_date and current_flag for each business record
UPDATE business_dim bd
SET bd.end_date = (
        SELECT CASE
```

```
            WHEN MAX(nc.contractEndDate) = TO_DATE('31/12/2023', 'DD/MM/YYYY') THEN NULL
            ELSE MAX(nc.contractEndDate)
        END
        FROM new_contract nc
        WHERE nc.stallID = bd.stallID
        AND nc.tenantID = bd.tenantID
    ),
    bd.current_flag = (
        SELECT CASE
            WHEN MAX(nc.contractEndDate) = TO_DATE('31/12/2023', 'DD/MM/YYYY') THEN 'Y'
            ELSE 'N'
        END
        FROM new_contract nc
        WHERE nc.stallID = bd.stallID
        AND nc.tenantID = bd.tenantID
    );
```

Customer Dimension Table

```
INSERT INTO Customer_dim (
    CUSTOMER_KEY,
    CUSTOMER_ID,
    CUSTOMER_BIRTHDATE,
    CUSTOMER_GENDER
)
SELECT
    cust_dim_seq.NEXTVAL,
    CUSTID,
    CUSTBIRTHDATE,
    CUSTGENDER
FROM new_cust;


-- Default (dine in)
UPDATE customer_dim
SET customer_key = -1
WHERE customer_key = 100007;
```

Platform Dimension Table

```
CREATE TABLE Platform_dim (
    platform_key NUMBER  not null,
    platformID varchar(5) NOT NULL,
    deliveryID NUMBER not null,
    PlatformName varchar(10) NOT NULL,
    DriverName varchar(20) not null,
```

```
    PickUpTime varchar(10) not null,
    ArrivedTime varchar(10) not null,
    DeliveryCost number(4,2) not null,
    DeliveryAddress varchar(50) not null,
CONSTRAINT PK_Platform_dim PRIMARY KEY(platform_key)
);
```

Menu Dimension Table

```
drop sequence menu_dim_seq;
CREATE sequence menu_dim_seq
 start with 100001
 increment by 1;

INSERT INTO MENU_DIM (
    MENU_KEY,
    menuID,
    FOODNDRINK,
    ITEM_DESC,
    ITEM_PRICE
)
SELECT
    menu_dim_seq.NEXTVAL,
    menuID,
    foodNDrink,
    ItemDesc,
    ItemPrice
FROM new_menu;
```

Contract Fact Table

```
INSERT INTO contract_fact
SELECT
    D.date_key,           -- Date dimension key
    B.business_key,      -- Business dimension key
    A.contractID,
    A.contractStartDate,
    A.contractEndDate,
    A.penaltyAmount
FROM new_contract A
JOIN date_dim D ON A.contractStartDate = D.cal_date
JOIN business_dim B ON A.stallID = B.stallID
WHERE B.tenantID = A.tenantID;
```

Order Fact Table

```
INSERT INTO Order_Facts (
    CUSTOMER_KEY, platform_key, Menu_key, Business_key, DATE_KEY, orderID, quantity, OrderRating,
unit_price
)SELECT
    (CASE
        WHEN A.customerID IS NULL THEN -1   -- If customerID is NULL, assign -1 to customer_key
        ELSE D.customer_key                 -- Otherwise, use the actual customer_key
     END) AS customer_key,
    (CASE
        WHEN A.deliveryID IS NULL THEN -1   -- If deliveryID is NULL, assign -1 to plat_key
        ELSE E.Platform_key                     -- Otherwise, use the actual plat_key
     END) AS Platform_key,
    F.MENU_KEY AS menu_key,
    H.Business_key AS business_key,
    C.DATE_KEY AS date_key,
    A.orderID AS orderID,
    B.quantity AS quantity,
    A.rating AS OrderRating,
    F.ITEM_PRICE AS unit_price
FROM new_orders A
JOIN new_order_details B ON A.orderID = B.orderID
JOIN Date_dim C ON A.orderDate = C.cal_date
LEFT JOIN customer_dim D ON A.customerID = D.customer_ID  -- LEFT JOIN for potential nulls
LEFT JOIN Platform_dim E ON A.deliveryID = E.deliveryID  -- LEFT JOIN for potential nulls
JOIN Menu_dim F ON B.menuID = F.menuID
JOIN new_menu G ON F.menuID = G.menuID
JOIN Business_dim H ON G.stallID = H.stallID
WHERE
    H.start_date <= C.cal_date AND
    (H.end_date IS NULL OR H.end_date >= C.cal_date)
    AND EXTRACT(YEAR FROM C.cal_date) BETWEEN 2014 AND 2023;
```

2.2 Script for subsequent loading

Date Dimension Table

```
CREATE OR REPLACE PROCEDURE Prod_Insert_Date_Dim AS
    startDate           DATE := TO_DATE('01/01/2014', 'DD/MM/YYYY');
    endDate             DATE := TO_DATE('31/12/2023', 'DD/MM/YYYY');
    v_CAL_DATE          DATE;
    v_FULL_DESC         VARCHAR2(40);
    v_DAY_WEEK          NUMBER(1);
    v_DAY_NUM_MONTH     NUMBER(2);
    v_DAY_NUM_YEAR      NUMBER(3);
    v_LAST_DAY_IND      CHAR(1);
    v_CAL_WEEK_END_DATE DATE;
    v_CAL_WEEK_YEAR     NUMBER(2);
    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_HOLIDAY_IND       CHAR(1);
    v_WEEKDAY_IND       CHAR(1);
    v_FESTIVE_EVENT     VARCHAR2(25);
    v_count             NUMBER;
BEGIN
    WHILE (startDate <= endDate) LOOP
        v_CAL_DATE := startDate;
        v_FULL_DESC := SUBSTR(TO_CHAR(startDate, 'Year') || ' ' || TO_CHAR(startDate, 'MONTH') ||
' ' || TO_CHAR(startDate, 'DD'), 1, 31);
        v_DAY_WEEK := TO_CHAR(startDate, 'D');
        v_DAY_NUM_MONTH := TO_CHAR(startDate, 'DD');
        v_DAY_NUM_YEAR := TO_CHAR(startDate, 'DDD');

        IF (startDate = LAST_DAY(startDate)) THEN
            v_LAST_DAY_IND := 'Y';
        ELSE
            v_LAST_DAY_IND := 'N';
        END IF;

        v_CAL_WEEK_END_DATE := startDate + (7 - v_DAY_WEEK);
        v_CAL_WEEK_YEAR := TO_CHAR(startDate, 'IW');
        v_MONTH_NAME := TO_CHAR(startDate, 'MONTH');
        v_CAL_MONTH_YEAR := TO_CHAR(startDate, 'MM');
        v_CAL_YEAR_MONTH := TO_CHAR(startDate, 'YYYY-MM');
        v_CAL_QUARTER := TO_CHAR(startDate, 'Q');
```

```
v_CAL_YEAR_QUARTER := TO_CHAR(startDate, 'YYYY-Q');
v_CAL_YEAR := TO_CHAR(startDate, 'YYYY');
v_HOLIDAY_IND := 'N';

IF (v_DAY_WEEK BETWEEN 2 AND 6) THEN
    v_WEEKDAY_IND := 'Y';
ELSE
    v_WEEKDAY_IND := 'N';
END IF;

v_FESTIVE_EVENT := NULL;

-- Check if the date already exists in date_dim
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM date_dim
    WHERE cal_date = v_CAL_DATE;

    IF v_count = 0 THEN
        INSERT INTO date_dim (
            date_key, cal_date, full_desc, day_week, day_num_month,
            day_num_year, last_day_ind, cal_week_end_date, cal_week_year,
            month_name, cal_month_year, cal_year_month, cal_quarter,
            cal_year_quarter, cal_year, holiday_ind, weekday_ind, festive_event
        ) VALUES (
            date_seq.NEXTVAL,
            v_CAL_DATE,
            v_FULL_DESC,
            v_DAY_WEEK,
            v_DAY_NUM_MONTH,
            v_DAY_NUM_YEAR,
            v_LAST_DAY_IND,
            v_CAL_WEEK_END_DATE,
            v_CAL_WEEK_YEAR,
            v_MONTH_NAME,
            v_CAL_MONTH_YEAR,
            v_CAL_YEAR_MONTH,
            v_CAL_QUARTER,
            v_CAL_YEAR_QUARTER,
            v_CAL_YEAR,
            v_HOLIDAY_IND,
            v_WEEKDAY_IND,
            v_FESTIVE_EVENT
        );
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        NULL;
```

```
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('Error checking or inserting date: ' || SQLERRM);
        END;

        startDate := startDate + 1;
    END LOOP;
END Prod_Insert_Date_Dim;
/
EXEC Prod_Insert_Date_Dim;
```

Business Dimension Table

```
INSERT INTO business_dim (
    business_key,
    stallID,
    tenantID,
    tenantName,
    tenantEmail,
    stallSize,
    stallType,
    stallName
)
SELECT
    business_dim_seq.nextval,
    S.stallID,
    T.tenantID,
    T.tenantPerson,
    T.tenantEmail,
    S.stallSize,
    S.stallType,
    C.stallName
FROM
    new_stall S
JOIN
    new_contract C ON S.stallID = C.stallID
JOIN
    new_tenants T ON C.tenantID = T.tenantID
WHERE
    (S.stallID, T.tenantID) NOT IN (
        SELECT BD.stallID, BD.tenantID
        FROM business_dim BD
    );
```

Customer Dimension Table

```
INSERT INTO Customer_dim (
    CUSTOMER_KEY,
    CUSTOMER_ID,
    CUSTOMER_BIRTHDATE,
    CUSTOMER_GENDER
)
SELECT
    cust_dim_seq.NEXTVAL,
    nc.CUSTID,
    nc.CUSTBIRTHDATE,
    nc.CUSTGENDER
FROM new_cust nc
WHERE nc.CUSTID NOT IN (
    SELECT CUSTOMER_ID
    FROM Customer_dim
);
```

Platform Dimension Table

```
INSERT INTO platform_dim
SELECT platform_dim_seq.nextval,
       P.platformID,
       D.deliveryID,
       P.PlatformName,
       D.DriverName,
       D.PickUpTime,
       D.ArrivedTime,
       D.DeliveryCost,
       D.DeliveryAddress
FROM new_platform P
JOIN new_delivery D ON P.platformID = D.platformID
WHERE (P.platformID, D.deliveryID) NOT IN (
    SELECT platformID, deliveryID
    FROM platform_dim
);
```

Menu Dimension Table

```
INSERT INTO MENU_DIM (
    MENU_KEY,
    menuID,
    FOODNDRINK,
    ITEM_DESC,
    ITEM_PRICE
)
```

```
SELECT
    menu_dim_seq.NEXTVAL,
    nm.menuID,
    nm.foodNDrink,
    nm.ItemDesc,
    nm.ItemPrice
FROM new_menu nm
WHERE nm.menuID NOT IN (SELECT md.menuID FROM MENU_DIM md);
```

Contract Fact Table

```
INSERT INTO contract_fact (
    date_key,
    business_key,
    contractID,
    contractStartDate,
    contractEndDate,
    penaltyAmount
)
SELECT
    B.date_key,
    C.business_key,
    A.contractID,
    A.contractStartDate,
    A.contractEndDate,
    A.penaltyAmount
FROM
    new_contract A
JOIN
    date_dim B ON A.contractStartDate = B.cal_date
JOIN
    business_dim C ON A.stallID = C.stallID
WHERE
    A.contractID NOT IN (
        SELECT CF.contractID
        FROM contract_fact CF
    );
```

Order Fact Table

```
INSERT INTO Order_Facts (
    CUSTOMER_KEY, Platform_key, Menu_key, Business_key, DATE_KEY, orderID, quantity, OrderRating,
unit_price
)
```

```
SELECT
    D.Customer_key,
    E.Platform_key,
    F.MENU_KEY,
    H.Business_key,
    C.DATE_KEY,
    A.orderID,
    B.quantity,
    A.rating AS OrderRating,
    F.ITEM_PRICE AS unit_price
FROM
    new_orders A
JOIN
    new_order_details B ON A.orderID = B.orderID
JOIN
    Date_dim C ON A.orderDate = C.cal_date
JOIN
    customer_dim D ON A.customerID = D.customer_ID
JOIN
    Platform_dim E ON A.deliveryID = E.deliveryID
JOIN
    Menu_dim F ON B.menuID = F.menuID
JOIN
    new_menu G ON F.menuID = G.menuID
JOIN
    Business_dim H ON G.stallID = H.stallID
WHERE
    A.orderID NOT IN (
        SELECT orderID
        FROM Order_Facts
    );
```

2.3 Script for updating (Type 2 changes)

2.3.1 Update Status

```
-- Mark existing records as historical if business details have changed, using NOT IN
UPDATE business_dim bd
SET bd.end_date = SYSDATE,
    bd.current_flag = 'N'
WHERE bd.current_flag = 'Y'
AND (bd.stallID, bd.tenantID) IN (
    SELECT S.stallID, T.tenantID
    FROM new_stall S
    JOIN new_contract C ON S.stallID = C.stallID
    JOIN new_tenants T ON C.tenantID = T.tenantID
    WHERE (S.stallID, T.tenantID) NOT IN (
        SELECT bd2.stallID, bd2.tenantID
        FROM business_dim bd2
        WHERE bd2.stallID = S.stallID
        AND bd2.tenantID = T.tenantID
        AND bd2.tenantName = T.tenantPerson
        AND bd2.tenantEmail = T.tenantEmail
        AND bd2.stallSize = S.stallSize
        AND bd2.stallType = S.stallType
        AND bd2.stallName = C.stallName
    )
);
```

2.3.2 Insert New Row

```
-- Insert new version of changed records using NOT IN
INSERT INTO business_dim (
    business_key, stallID, tenantID, tenantName, tenantEmail, stallSize, stallType, stallName,
start_date, current_flag
)
SELECT business_dim_seq.NEXTVAL, S.stallID, T.tenantID, T.tenantPerson, T.tenantEmail,
S.stallSize, S.stallType, C.stallName, SYSDATE, 'Y'
FROM new_stall S
JOIN new_contract C ON S.stallID = C.stallID
JOIN new_tenants T ON C.tenantID = T.tenantID
WHERE (S.stallID, T.tenantID) NOT IN (
    SELECT bd.stallID, bd.tenantID
```

```
    FROM business_dim bd
    WHERE bd.stallID = S.stallID
    AND bd.tenantID = T.tenantID
    AND bd.tenantName = T.tenantPerson
    AND bd.tenantEmail = T.tenantEmail
    AND bd.stallSize = S.stallSize
    AND bd.stallType = S.stallType
    AND bd.stallName = C.stallName
);
```

Chapter 3 Business Analytics Reports

3.1 Chong Xu Ming

3.1.1 Selected Stall Food Rating

```
SET SERVEROUTPUT ON;

set pagesize 120
Set linesize 300

DECLARE
    v_stall_id VARCHAR2(10);
    v_last_year NUMBER := -1;  -- Initialize with a non-null value that won't match any valid
year
    TYPE food_type IS TABLE OF NUMBER INDEX BY VARCHAR2(100);  -- Store previous year quantities
    v_prev_quantity food_type;  -- To store total quantities from the previous year
    v_change VARCHAR2(5);  -- Store change indicator
    v_percentage_change NUMBER;  -- To store the percentage change
BEGIN
    -- Title for the report

    -- Prompt user for input
    DBMS_OUTPUT.PUT_LINE('Choose the stall you want to check:');

    -- Accept input (this would typically come from a UI or application)
    v_stall_id := '&stall_id'; -- Accept user input

    -- Display chosen stall

DBMS_OUTPUT.PUT_LINE('************************************************************************');
    DBMS_OUTPUT.PUT_LINE('***               STALL FOOD RATING CHECK ON ORDERS
***');

DBMS_OUTPUT.PUT_LINE('************************************************************************');

    DBMS_OUTPUT.PUT_LINE('You selected Stall ID: ' || v_stall_id);

    -- Execute the query with the user's input
    FOR rec IN (
        SELECT
            dd.cal_year,
            md.FOODNDRINK,
            md.ITEM_PRICE,
            SUM(ofs.quantity) AS total_quantity,
            TO_CHAR(AVG(ofs.OrderRating), '999.9999') AS avg_rating  -- Format avg_rating to 4
decimal places
```

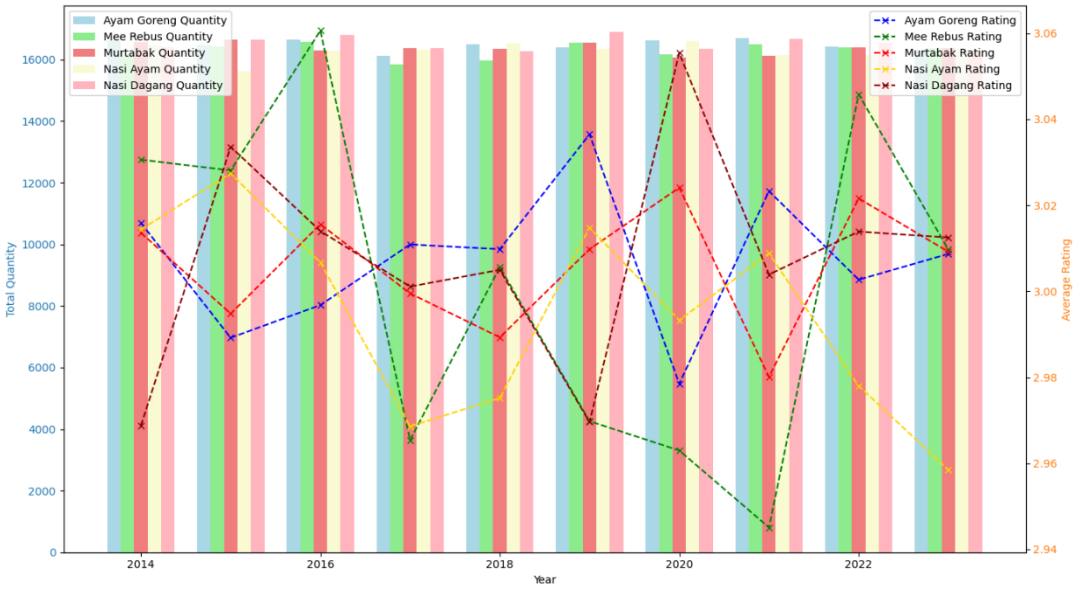
```
        FROM
            Order_Facts ofs
        JOIN
            MENU_DIM md ON ofs.Menu_key = md.MENU_KEY
        JOIN
            Business_Dim bd ON ofs.Business_key = bd.business_key
        JOIN
            Date_Dim dd ON ofs.DATE_KEY = dd.date_key
        WHERE
            bd.StallID = v_stall_id  -- Use the user input here
        GROUP BY
            dd.cal_year,
            md.FOODNDRINK,
            md.ITEM_PRICE
        ORDER BY
            dd.cal_year
    ) LOOP
        -- Display the year only once when it changes and display the header after the year
        IF rec.cal_year != v_last_year THEN
            DBMS_OUTPUT.PUT_LINE(CHR(10) || 'Year: ' || rec.cal_year);  -- Display the year

DBMS_OUTPUT.PUT_LINE('-----------------------------------------------------------------------');
            DBMS_OUTPUT.PUT_LINE('Food/Drink            | Price   | Total Quantity       | Avg
Rating');

DBMS_OUTPUT.PUT_LINE('-----------------------------------------------------------------------');
            v_last_year := rec.cal_year;  -- Update the last year to the current one
        END IF;

        -- Calculate the change in total quantity and the percentage change compared to the
previous year
        IF v_prev_quantity.EXISTS(rec.FOODNDRINK) THEN
            IF rec.total_quantity > v_prev_quantity(rec.FOODNDRINK) THEN
                v_change := '(+)';
                v_percentage_change := ((rec.total_quantity - v_prev_quantity(rec.FOODNDRINK)) /
v_prev_quantity(rec.FOODNDRINK)) * 100;
            ELSIF rec.total_quantity < v_prev_quantity(rec.FOODNDRINK) THEN
                v_change := '(-)';
                v_percentage_change := ((v_prev_quantity(rec.FOODNDRINK) - rec.total_quantity) /
v_prev_quantity(rec.FOODNDRINK)) * 100;
            ELSE
                v_change := '';  -- No change
                v_percentage_change := 0;
            END IF;
        ELSE
            v_change := '';  -- No previous year data for this food item
            v_percentage_change := NULL;  -- No percentage change can be calculated
        END IF;
```

```
        -- Display the data for each record with formatted output, the change indicator, and the
percentage
        IF v_percentage_change IS NOT NULL THEN
            DBMS_OUTPUT.PUT_LINE(RPAD(rec.FOODNDRINK, 20) || ' | ' ||
                                 LPAD(TO_CHAR(rec.ITEM_PRICE, '999.99'), 8) || ' | ' ||
                                 LPAD(TO_CHAR(rec.total_quantity, '9999999') || v_change || ' '
|| TO_CHAR(v_percentage_change, '990.00') || '%', 20) || ' | ' ||
                                 LPAD(rec.avg_rating, 10, ' '));
        ELSE
            DBMS_OUTPUT.PUT_LINE(RPAD(rec.FOODNDRINK, 20) || ' | ' ||
                                 LPAD(TO_CHAR(rec.ITEM_PRICE, '999.99'), 8) || ' | ' ||
                                 LPAD(TO_CHAR(rec.total_quantity, '9999999'), 20) || ' | ' ||
                                 LPAD(rec.avg_rating, 10, ' '));
        END IF;

        -- Store the current year's total quantity for comparison in the next iteration
        v_prev_quantity(rec.FOODNDRINK) := rec.total_quantity;
    END LOOP;

    -- If no records were found, display a message
    IF v_last_year = -1 THEN
        DBMS_OUTPUT.PUT_LINE('No records found for the selected stall.');
    END IF;
END;
/
```

```
Enter value for stall_id: S117
old  15:     v_stall_id := '&stall_id'; -- Accept user input
new  15:     v_stall_id := 'S117'; -- Accept user input
Choose the stall you want to check:
*********************************************************************
***              STALL FOOD RATING CHECK ON ORDERS                ***
*********************************************************************
You selected Stall ID: S117

Year: 2014
----------------------------------------------------------------------
Food/Drink           | Price    | Total Quantity       | Avg Rating
----------------------------------------------------------------------
Ayam Goreng          |    13.25 |              16620 |     3.0159
Mee Rebus            |    12.75 |              16171 |     3.0306
Murtabak             |     9.50 |              16582 |     3.0136
Nasi Ayam            |    11.00 |              16454 |     3.0144
Nasi Dagang          |    14.75 |              16397 |     2.9688

Year: 2015
----------------------------------------------------------------------
Food/Drink           | Price    | Total Quantity       | Avg Rating
----------------------------------------------------------------------
Ayam Goreng          |    13.25 |    16476(-)    0.87% |     2.9892
Mee Rebus            |    12.75 |    16419(+)    1.53% |     3.0281
Murtabak             |     9.50 |    16661(+)    0.48% |     2.9948
Nasi Ayam            |    11.00 |    15622(-)    5.06% |     3.0274
Nasi Dagang          |    14.75 |    16654(+)    1.57% |     3.0336

Year: 2016
----------------------------------------------------------------------
Food/Drink           | Price    | Total Quantity       | Avg Rating
----------------------------------------------------------------------
Ayam Goreng          |    13.25 |    16642(+)    1.01% |     2.9968
Mee Rebus            |    12.75 |    16569(+)    0.91% |     3.0606
Murtabak             |     9.50 |    16297(-)    2.18% |     3.0156
Nasi Ayam            |    11.00 |    16295(+)    4.31% |     3.0067
Nasi Dagang          |    14.75 |    16796(+)    0.85% |     3.0140

Year: 2017
----------------------------------------------------------------------
Food/Drink           | Price    | Total Quantity       | Avg Rating
----------------------------------------------------------------------
Ayam Goreng          |    13.25 |    16116(-)    3.16% |     3.0109
Mee Rebus            |    12.75 |    15856(-)    4.30% |     2.9654
Murtabak             |     9.50 |    16364(+)    0.41% |     2.9995
Nasi Ayam            |    11.00 |    16332(+)    0.23% |     2.9686
Nasi Dagang          |    14.75 |    16363(-)    2.58% |     3.0011
```

```
Year: 2018
----------------------------------------------------------------------
Food/Drink           | Price    | Total Quantity       | Avg Rating
----------------------------------------------------------------------
Ayam Goreng          |    13.25 |    16511(+)    2.45% |     3.0098
Mee Rebus            |    12.75 |    15982(+)    0.79% |     3.0056
Murtabak             |     9.50 |    16354(-)    0.06% |     2.9893
Nasi Ayam            |    11.00 |    16531(+)    1.22% |     2.9752
Nasi Dagang          |    14.75 |    16267(-)    0.59% |     3.0050

Year: 2019
----------------------------------------------------------------------
Food/Drink           | Price    | Total Quantity       | Avg Rating
----------------------------------------------------------------------
Ayam Goreng          |    13.25 |    16388(-)    0.74% |     3.0365
Mee Rebus            |    12.75 |    16546(+)    3.53% |     2.9698
Murtabak             |     9.50 |    16537(+)    1.12% |     3.0098
Nasi Ayam            |    11.00 |    16353(-)    1.08% |     3.0148
Nasi Dagang          |    14.75 |    16902(+)    3.90% |     2.9696

Year: 2020
----------------------------------------------------------------------
Food/Drink           | Price    | Total Quantity       | Avg Rating
----------------------------------------------------------------------
Ayam Goreng          |    13.25 |    16632(+)    1.49% |     2.9785
Mee Rebus            |    12.75 |    16172(-)    2.26% |     2.9630
Murtabak             |     9.50 |    16073(-)    2.81% |     3.0241
Nasi Ayam            |    11.00 |    16598(+)    1.50% |     2.9933
Nasi Dagang          |    14.75 |    16355(-)    3.24% |     3.0554

Year: 2021
----------------------------------------------------------------------
Food/Drink           | Price    | Total Quantity       | Avg Rating
----------------------------------------------------------------------
Ayam Goreng          |    13.25 |    16703(+)    0.43% |     3.0233
Mee Rebus            |    12.75 |    16508(+)    2.08% |     2.9451
Murtabak             |     9.50 |    16116(+)    0.27% |     2.9801
Nasi Ayam            |    11.00 |    16137(-)    2.78% |     3.0088
Nasi Dagang          |    14.75 |    16675(+)    1.96% |     3.0039
```

```
YEAR: 2022 | QUARTER  | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS | AVG WEEKEND ORDERS  | AVG ORDER DIFFERENCE
-------------------------------------------------------------------------------------------------------------------------
Q1         |          14512 |          5845 |              3.02 |              2.99 |                  .03
Q2         |          14720 |          5730 |              2.99 |              2.96 |                  .03
Q3         |          14609 |          5730 |              2.99 |              2.96 |                  .03
Q4         |          14673 |          6095 |              3.02 |              3.03 |                 -.01

YEAR: 2023 | QUARTER  | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS | AVG WEEKEND ORDERS  | AVG ORDER DIFFERENCE
-------------------------------------------------------------------------------------------------------------------------
Q1         |          14452 |          5549 |              2.99 |              3.00 |                 -.01
Q2         |          14974 |          5780 |              3.02 |              2.98 |                  .05
Q3         |          14767 |          5830 |              3.02 |              3.03 |                 -.01
Q4         |          14320 |          5973 |              2.94 |              3.03 |                 -.08

PL/SQL procedure successfully completed.
```



Based on the report and plot we can choose the specific stall and see the menu rating and its total order quantity in the decision making . By looking at the ayam goreng of year 2014 to 2015 when the price is not change but the total quantity is decreasing and the rating is also decreasing , at this point we can further think that is the food is become not so good maybe is because the chef is not so good or the food quality is not so good , and for murtabak the price remain the same , total order quantity is increasing but the order rating is having a bit of decreasing in this situation we can predict that maybe just not suitable for the flavor for others but the price is cheap so they decided to buy this meal.

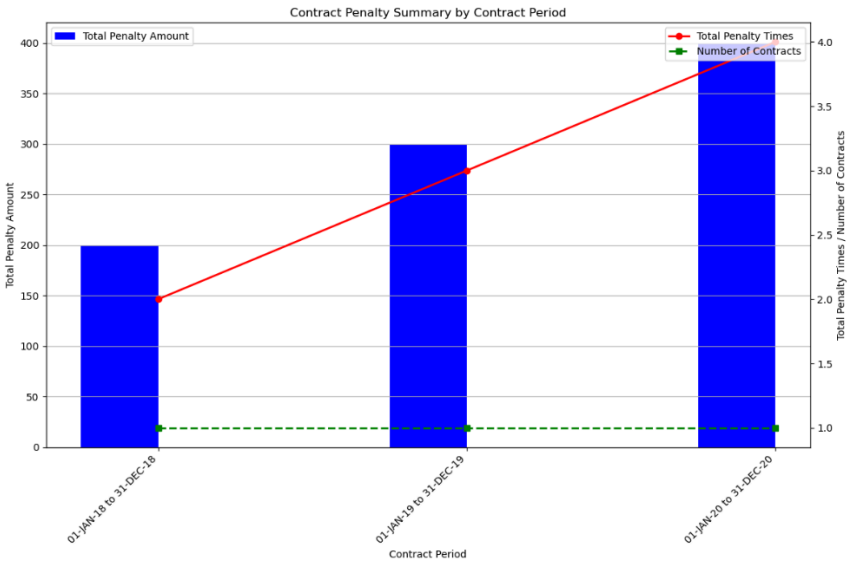
3.1.2 Selected Tenant Contract Checking

```
SET SERVEROUTPUT ON;

set pagesize 120
Set linesize 300

DECLARE
    v_tenant_id VARCHAR2(10);
    v_tenant_name VARCHAR2(100);
BEGIN


    -- Prompt user for tenant ID
    DBMS_OUTPUT.PUT_LINE('Enter the tenantID you want to check:');

    -- Accept user input for tenant ID
    v_tenant_id := '&tenantID';  -- Prompt for input; this will be replaced by the user's choice

    -- Retrieve the tenant name based on the selected tenant ID
    SELECT bd.tenantName
    INTO v_tenant_name
    FROM business_dim bd
    WHERE bd.tenantID = v_tenant_id;
```

```
      -- Title for the report

DBMS_OUTPUT.PUT_LINE('********************************************************************************
******************');
    DBMS_OUTPUT.PUT_LINE('***                                  CONTRACT PENALTY CHECK
***');

DBMS_OUTPUT.PUT_LINE('********************************************************************************
******************');

    -- Display chosen tenant ID and tenant name
    DBMS_OUTPUT.PUT_LINE('You selected Tenant ID: ' || v_tenant_id || ' (' || v_tenant_name ||
')');

    -- Display the headers with fixed widths
    DBMS_OUTPUT.PUT_LINE(RPAD('Contract Period', 25) || ' | ' ||
                         RPAD('Total Penalty Amount', 20) || ' | ' ||
                         RPAD('Total Penalty Times', 20) || ' | ' ||
                         'Number of Contracts');

DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------------------------
------------------');

    -- Execute the query with the user's input
    FOR rec IN (
        SELECT
            cf.contractStartDate || ' to ' || cf.contractEndDate AS contract_period,  -- Combine
dates
            SUM(cf.penaltyAmount) AS total_penalty_amount,
            SUM(cf.penaltyAmount / 100) AS total_penalty_times,
            COUNT(cf.contractID) AS number_of_contracts
        FROM
            contract_fact cf
        JOIN
            business_dim bd ON cf.business_key = bd.business_key
        WHERE
            bd.tenantID = v_tenant_id  -- Use the user input here
        GROUP BY
            cf.contractStartDate,
            cf.contractEndDate
        ORDER BY
            cf.contractStartDate
    ) LOOP
        -- Display the result with aligned output
        DBMS_OUTPUT.PUT_LINE(RPAD(rec.contract_period, 25) || ' | ' ||
                             LPAD(rec.total_penalty_amount, 20) || ' | ' ||
                             LPAD(rec.total_penalty_times, 20) || ' | ' ||
                             LPAD(rec.number_of_contracts, 18));
```

```
    END LOOP;

    -- Check if no records were found
    IF NOT SQL%ROWCOUNT > 0 THEN
        DBMS_OUTPUT.PUT_LINE('No records found for the selected tenant.');
    END IF;

EXCEPTION
    -- Handle case where the tenant ID does not exist
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No tenant found with the provided tenant ID.');
END;
/
```

```
Enter value for tenantid: T121
old   9:     v_tenant_id := '&tenantID';  -- Prompt for input; this will be replaced by the user's choice
new   9:     v_tenant_id := 'T121';  -- Prompt for input; this will be replaced by the user's choice
********************************************************************************************
***                              CONTRACT PENALTY CHECK                                  ***
********************************************************************************************
You selected Tenant ID: T121 (Bob Wang)
Contract Period          | Total Penalty Amount | Total Penalty Times  | Number of Contracts
-------------------------------------------------------------------------------------------
01-JAN-18 to 31-DEC-18   |                  200 |                    2 |                   1
01-JAN-19 to 31-DEC-19   |                  300 |                    3 |                   1
01-JAN-20 to 31-DEC-20   |                  400 |                    4 |                   1

PL/SQL procedure successfully completed.
```



Based on this report and plot we want to find out that whenever a tenant wants to continue its contract do we want to continue with this tenant based on his previous amount times and total amount. For example we can see that tenant Bob Wang its total amount and its penalty times is increasing for the past few years and maybe will decide not to continue the contract with him anymore

3.1.3 Quarterly Order Summary for Selected Stall

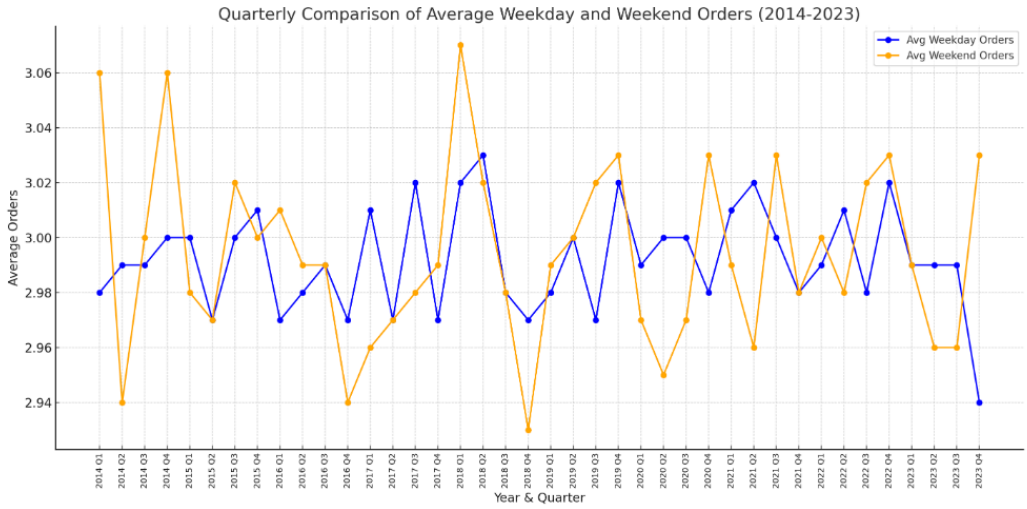
```
SET SERVEROUTPUT ON;
set pagesize 120
Set linesize 300
DECLARE
    v_stallID VARCHAR2(4);
    v_current_year NUMBER := 0;  -- Variable to track the current year
    v_stallName VARCHAR2(50);  -- Variable to store stall name
    v_stallName_displayed BOOLEAN := FALSE; -- Flag to display the stall name only once
BEGIN
    -- Prompt the user to input the stallID
    DBMS_OUTPUT.PUT_LINE('Enter the stallID you want to check:');
    v_stallID := '&enter_stallID';

    -- Print the title
    DBMS_OUTPUT.PUT_LINE('-------------------------------------------');
    DBMS_OUTPUT.PUT_LINE('Quarterly Order Summary for Selected Stall   ');
    DBMS_OUTPUT.PUT_LINE('-------------------------------------------');

    -- Query to get order counts for weekdays and weekends aggregated quarterly with averages per
day
    FOR rec IN (
        SELECT
            EXTRACT(YEAR FROM dd.cal_date) AS year,
            'Q' || TO_CHAR(dd.cal_date, 'Q') AS quarter,  -- Get quarter number (1 to 4)
            bd.stallName,
            SUM(CASE WHEN dd.weekDay_ind = 'Y' THEN ofs.quantity ELSE 0 END) AS weekday_orders,
            SUM(CASE WHEN dd.weekDay_ind = 'N' THEN ofs.quantity ELSE 0 END) AS weekend_orders,
            -- Count the number of weekdays and weekend days in the quarter
            SUM(CASE WHEN dd.weekDay_ind = 'Y' THEN 1 ELSE 0 END) AS num_weekdays,
            SUM(CASE WHEN dd.weekDay_ind = 'N' THEN 1 ELSE 0 END) AS num_weekend_days
        FROM
            Order_Facts ofs
        JOIN
            Business_Dim bd ON ofs.Business_key = bd.business_key
        JOIN
            Date_Dim dd ON ofs.DATE_KEY = dd.date_key
        WHERE
            bd.StallID = v_stallID
        GROUP BY
            EXTRACT(YEAR FROM dd.cal_date),
            TO_CHAR(dd.cal_date, 'Q'),  -- Group by quarter
            bd.stallName
        ORDER BY
            year, quarter
    )
    LOOP
        -- Print the stall name only once at the top (before the year and header)
        IF NOT v_stallName_displayed THEN
```

```
            v_stallName := rec.stallName;
            DBMS_OUTPUT.PUT_LINE('Stall Name: ' || v_stallName);  -- Display stall name once
            v_stallName_displayed := TRUE;  -- Set flag to true so the stall name is not printed
again
        END IF;

        -- Check if the year has changed
        IF rec.year != v_current_year THEN
            -- Update the current year
            v_current_year := rec.year;

            -- Display a new header for the new year
            DBMS_OUTPUT.PUT_LINE(CHR(10) || '------------------------'); -- Adds spacing before
new year table
            DBMS_OUTPUT.PUT_LINE('YEAR: ' || v_current_year);
            DBMS_OUTPUT.PUT_LINE('------------------------');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
-----------------------------------------------------');
            DBMS_OUTPUT.PUT_LINE('QUARTER  | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS
| AVG WEEKEND ORDERS  | AVG ORDER DIFFERENCE');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
-----------------------------------------------------');
        END IF;

        -- Calculate average orders per day, handling possible division by zero
        DBMS_OUTPUT.PUT_LINE(
            RPAD(rec.quarter, 9) || ' | ' ||
            LPAD(TO_CHAR(rec.weekday_orders, '9999999'), 14) || ' | ' ||
            LPAD(TO_CHAR(rec.weekend_orders, '9999999'), 14) || ' | ' ||
            LPAD(TO_CHAR(CASE WHEN rec.num_weekdays > 0 THEN rec.weekday_orders /
rec.num_weekdays ELSE 0 END, '9999999.99'), 18) || ' | ' ||
            LPAD(TO_CHAR(CASE WHEN rec.num_weekend_days > 0 THEN rec.weekend_orders /
rec.num_weekend_days ELSE 0 END, '9999999.99'), 18) || ' | ' ||
            LPAD(TO_CHAR(
                CASE
                    WHEN rec.num_weekdays > 0 AND rec.num_weekend_days > 0 THEN
                        (rec.weekday_orders / rec.num_weekdays) - (rec.weekend_orders /
rec.num_weekend_days)
                    ELSE 0
                END, '9999999.99'), 22)
        );
    END LOOP;
END;
/
```

```
Enter value for enter_stallid: S117
old   9:     v_stallID := '&enter_stallID';
new   9:     v_stallID := 'S117';
Enter the stallID you want to check:
**************************************************************************************************
*****                 Quarterly Order Summary for Selected Stall                              *****
**************************************************************************************************
Stall Name: Flavorful Bites

------------------------
YEAR: 2014
------------------------
-------------------------------------------------------------------------------------------------------
QUARTER  | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS | AVG WEEKEND ORDERS  | AVG ORDER DIFFERENCE
-------------------------------------------------------------------------------------------------------
Q1       |          14185 |           5882 |               2.98 |               3.06 |                -.08
Q2       |          14488 |           5755 |               2.99 |               2.94 |                 .05
Q3       |          15061 |           5868 |               2.99 |               3.00 |                -.01
Q4       |          14844 |           6141 |               3.00 |               3.06 |                -.07

------------------------
YEAR: 2015
------------------------
-------------------------------------------------------------------------------------------------------
QUARTER  | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS | AVG WEEKEND ORDERS  | AVG ORDER DIFFERENCE
-------------------------------------------------------------------------------------------------------
Q1       |          14174 |           5736 |               3.00 |               2.98 |                 .02
Q2       |          14603 |           5983 |               2.97 |               2.97 |                 .00
Q3       |          14804 |           5998 |               3.00 |               3.02 |                -.03
Q4       |          14736 |           5798 |               3.01 |               3.00 |                 .01

------------------------
YEAR: 2016
------------------------
-------------------------------------------------------------------------------------------------------
QUARTER  | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS | AVG WEEKEND ORDERS  | AVG ORDER DIFFERENCE
-------------------------------------------------------------------------------------------------------
Q1       |          14313 |           6068 |               2.97 |               3.01 |                -.04
Q2       |          14867 |           5766 |               2.98 |               2.96 |                 .02
Q3       |          15061 |           5917 |               2.99 |               2.99 |                 .00
Q4       |          14526 |           6081 |               2.97 |               2.94 |                 .03
```

```
------------------------
YEAR: 2017
------------------------
-------------------------------------------------------------------------------------------------------
QUARTER  | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS | AVG WEEKEND ORDERS  | AVG ORDER DIFFERENCE
-------------------------------------------------------------------------------------------------------
Q1       |          14948 |           5481 |               3.01 |               2.96 |                 .05
Q2       |          14157 |           5733 |               2.97 |               2.97 |                 .00
Q3       |          14508 |           5959 |               3.02 |               2.98 |                 .03
Q4       |          14457 |           5788 |               2.97 |               2.99 |                -.02

------------------------
YEAR: 2018
------------------------
-------------------------------------------------------------------------------------------------------
QUARTER  | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS | AVG WEEKEND ORDERS  | AVG ORDER DIFFERENCE
-------------------------------------------------------------------------------------------------------
Q1       |          14554 |           5531 |               3.02 |               3.07 |                -.05
Q2       |          14679 |           5535 |               3.03 |               3.02 |                 .01
Q3       |          14907 |           6323 |               2.98 |               2.98 |                 .00
Q4       |          14613 |           5503 |               2.97 |               2.93 |                 .04

------------------------
YEAR: 2019
------------------------
-------------------------------------------------------------------------------------------------------
QUARTER  | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS | AVG WEEKEND ORDERS  | AVG ORDER DIFFERENCE
-------------------------------------------------------------------------------------------------------
Q1       |          14456 |           5857 |               2.98 |               2.99 |                -.01
Q2       |          14735 |           5933 |               3.00 |               3.01 |                -.02
Q3       |          15070 |           5745 |               3.00 |               2.98 |                 .02
Q4       |          14918 |           6012 |               3.00 |               3.01 |                -.01

------------------------
YEAR: 2020
------------------------
-------------------------------------------------------------------------------------------------------
QUARTER  | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS | AVG WEEKEND ORDERS  | AVG ORDER DIFFERENCE
-------------------------------------------------------------------------------------------------------
Q1       |          14639 |           5797 |               3.01 |               3.05 |                -.05
Q2       |          14635 |           5692 |               3.03 |               2.95 |                 .08
Q3       |          13945 |           5870 |               3.00 |               3.02 |                -.02
Q4       |          15110 |           6142 |               2.98 |               3.01 |                -.03
```

```
-----------------------
YEAR: 2021
-----------------------
-------------------------------------------------------------------------------------------------------------
QUARTER   | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS | AVG WEEKEND ORDERS   | AVG ORDER DIFFERENCE
-------------------------------------------------------------------------------------------------------------
Q1        |          14147 |           5812 |               2.99 |                 2.97 |                 .02
Q2        |          14649 |           5792 |               2.98 |                 2.95 |                 .03
Q3        |          15098 |           5666 |               3.00 |                 2.97 |                 .03
Q4        |          15005 |           5970 |               2.99 |                 3.03 |                -.04

-----------------------
YEAR: 2022
-----------------------
-------------------------------------------------------------------------------------------------------------
QUARTER   | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS | AVG WEEKEND ORDERS   | AVG ORDER DIFFERENCE
-------------------------------------------------------------------------------------------------------------
Q1        |          14512 |           5845 |               3.02 |                 2.99 |                 .03
Q2        |          14720 |           5730 |               2.99 |                 2.96 |                 .03
Q3        |          14609 |           5730 |               2.99 |                 2.96 |                 .03
Q4        |          14673 |           6095 |               3.02 |                 3.03 |                -.01

-----------------------
YEAR: 2023
-----------------------
-------------------------------------------------------------------------------------------------------------
QUARTER   | WEEKDAY ORDERS | WEEKEND ORDERS | AVG WEEKDAY ORDERS | AVG WEEKEND ORDERS   | AVG ORDER DIFFERENCE
-------------------------------------------------------------------------------------------------------------
Q1        |          14452 |           5549 |               2.99 |                 3.00 |                -.01
Q2        |          14974 |           5780 |               3.02 |                 2.98 |                 .05
Q3        |          14767 |           5830 |               3.02 |                 3.03 |                -.01
Q4        |          14320 |           5973 |               2.94 |                 3.03 |                -.08

PL/SQL procedure successfully completed.
```



For this plot and table we decide to compare the difference of average weekday and weekend to see that is there need to more people to hired if the days is more average order and for the result we can see there is only a + - in between 0 to 1 so means that there is no big difference to hired people and reason why is using quarterly is because hired a people minimum requirement for most is 3 months.

3.2 Ten Wei Kang

3.2.1 Sales Growth Query

```
SET SERVEROUTPUT ON;
SET VERIFY off;
SET FEEDBACK off;
SET LINESIZE 150;
SET PAGESIZE 120;

PROMPT
PROMPT
PROMPT ~ Sales Growth Report ~

CREATE OR REPLACE PROCEDURE revenue_growth (start_year NUMBER, end_year NUMBER) AS
  CURSOR c_revenue IS
    WITH cte_revenue_data AS (
      SELECT
        D.cal_year AS Year,
        D.cal_quarter AS Quarter,
        SUM(O.quantity * O.unit_price) AS Current_Revenue
      FROM Order_Facts O
      JOIN Date_dim D ON O.Date_key = D.Date_key
      WHERE D.cal_year BETWEEN start_year AND end_year
      GROUP BY D.cal_year, D.cal_quarter
      ORDER BY D.cal_year, D.cal_quarter
    ),
    formatted_data AS (
      SELECT
        Year,
        Quarter,
        Current_Revenue,
        LAG(Current_Revenue) OVER (PARTITION BY Year ORDER BY Quarter) AS Previous_Revenue,
        ROUND(Current_Revenue - LAG(Current_Revenue) OVER (PARTITION BY Year ORDER BY Quarter),
2) AS Difference,
        CASE
          WHEN LAG(Current_Revenue) OVER (PARTITION BY Year ORDER BY Quarter) IS NOT NULL THEN
            ROUND(((Current_Revenue - LAG(Current_Revenue) OVER (PARTITION BY Year ORDER BY
Quarter)) /
                   LAG(Current_Revenue) OVER (PARTITION BY Year ORDER BY Quarter)) * 100, 2)
          ELSE NULL
        END AS Revenue_Growth_Percentage
      FROM cte_revenue_data
    )
    SELECT Year, Quarter, Current_Revenue, Previous_Revenue, Difference,
Revenue_Growth_Percentage
    FROM formatted_data;

  v_year NUMBER := NULL;
```

```
  v_curr_rev NUMBER;
  v_prev_rev NUMBER;
  v_diff NUMBER;
  v_growth_pct NUMBER;

  v_tot_rev NUMBER := 0;
  v_tot_growth NUMBER := 0;
  v_count_growth NUMBER := 0;

  v_overall_tot_rev NUMBER := 0;
  v_overall_growth NUMBER := 0;
  v_overall_count NUMBER := 0;

  TYPE Yearly_Sales IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
  yearly_totals Yearly_Sales;

BEGIN
    DBMS_OUTPUT.PUT_LINE(CHR(13));

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE('Sales Growth Report');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE(CHR(13));

  FOR rec IN c_revenue LOOP
    -- Check if the year has changed
    IF v_year IS NULL OR v_year != rec.Year THEN
      -- Output totals for the previous year
      IF v_year IS NOT NULL THEN

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
-------------');
        DBMS_OUTPUT.PUT_LINE('| Total revenue for year ' || v_year || ' : ' ||
RPAD(TO_CHAR(v_tot_rev, 'FM999G999G999D00'), 54) || ' |');
        IF yearly_totals.EXISTS(v_year - 1) THEN
          v_growth_pct := ((v_tot_rev - yearly_totals(v_year - 1)) / yearly_totals(v_year - 1)) *
100;
          DBMS_OUTPUT.PUT_LINE('| Total revenue growth percentage for year: ' ||
RPAD(ROUND(v_growth_pct, 2) || '%', 42) || ' |');
        ELSE
          DBMS_OUTPUT.PUT_LINE('| Total revenue growth percentage for year: N/A' || RPAD(' ', 39)
|| ' |');
        END IF;

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
-------------');
```

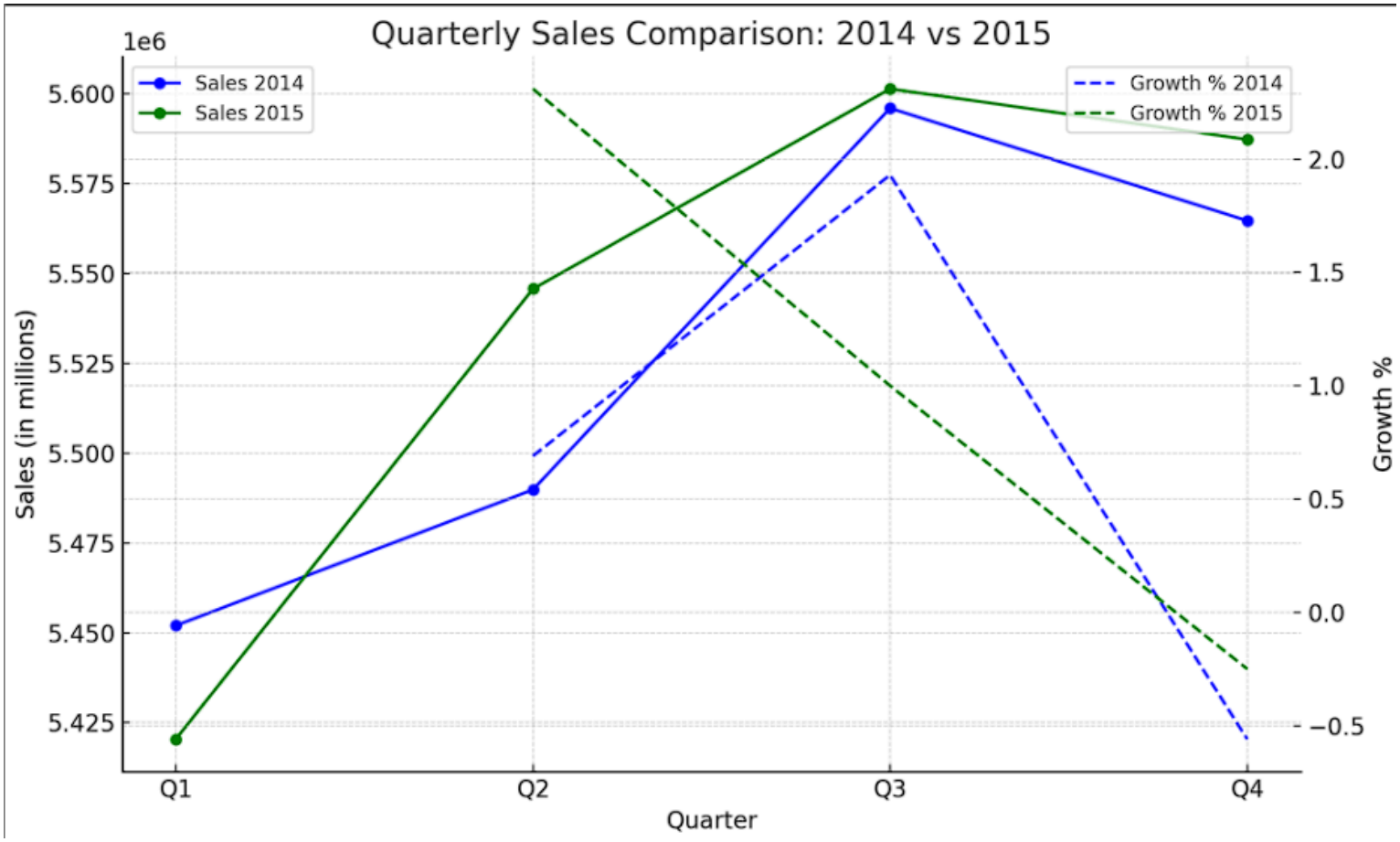
```
        -- Store yearly totals for final comparison
        yearly_totals(v_year) := v_tot_rev;
      END IF;

      -- Reset year-specific totals
      v_tot_rev := 0;
      v_tot_growth := 0;
      v_count_growth := 0;

      -- Set the new year
      v_year := rec.Year;
      DBMS_OUTPUT.PUT_LINE(CHR(13));
      DBMS_OUTPUT.PUT_LINE('Year: ' || v_year);


DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
-------------');
      DBMS_OUTPUT.PUT_LINE('| Quarter   |   Current Sales    |   Previous Sales  |    Difference
|   Growth %  |');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
-------------');
    END IF;

    -- Output quarterly data
    DBMS_OUTPUT.PUT_LINE('|   Q' || rec.Quarter || '      | ' ||
RPAD(TO_CHAR(rec.Current_Revenue, 'FM999G999G999D00'), 17) || ' | ' ||
                         RPAD(NVL(TO_CHAR(rec.Previous_Revenue, 'FM999G999G999D00'), '-'), 17) || '
| ' ||
                         RPAD(NVL(TO_CHAR(rec.Difference, 'FM999G999G999D00'), '-'), 17) || ' | ' ||
                         RPAD(NVL(TO_CHAR(rec.Revenue_Growth_Percentage, 'FM999D00'), '-'), 7) || '%
|');

    -- Update totals
    v_tot_rev := v_tot_rev + rec.Current_Revenue;
  END LOOP;

  -- Output totals for the last year
  IF v_year IS NOT NULL THEN

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
-------------');
    DBMS_OUTPUT.PUT_LINE('| Total revenue for year ' || v_year || ' : ' ||
RPAD(TO_CHAR(v_tot_rev, 'FM999G999G999D00'), 54) || ' |');
    IF yearly_totals.EXISTS(v_year - 1) THEN
      v_growth_pct := ((v_tot_rev - yearly_totals(v_year - 1)) / yearly_totals(v_year - 1)) *
100;
```

```
      DBMS_OUTPUT.PUT_LINE('| Total revenue growth percentage for year: ' ||
RPAD(ROUND(v_growth_pct, 2) || '%', 42) || ' |');
    ELSE
      DBMS_OUTPUT.PUT_LINE('| Total revenue growth percentage for year: N/A' || RPAD(' ', 39) ||
' |');
    END IF;

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------
-------------');

    -- Store yearly totals for final comparison
    yearly_totals(v_year) := v_tot_rev;
  END IF;

  -- Output overall summary: Comparison by Year
  DBMS_OUTPUT.PUT_LINE(CHR(13));
  DBMS_OUTPUT.PUT_LINE('Yearly Sales Comparison');

  DBMS_OUTPUT.PUT_LINE('-----------------------------------------------------------------');
  DBMS_OUTPUT.PUT_LINE('| Year     |    Total Sales      |  Growth from Previous Year |');
  DBMS_OUTPUT.PUT_LINE('-----------------------------------------------------------------');

  -- Iterate through yearly totals and print the comparison
  FOR i IN yearly_totals.FIRST..yearly_totals.LAST LOOP
    IF yearly_totals.EXISTS(i) THEN
      IF yearly_totals.EXISTS(i - 1) THEN
        DBMS_OUTPUT.PUT_LINE('| ' || i || '     | ' || RPAD(TO_CHAR(yearly_totals(i),
'FM999G999G999D00'), 19) || ' | ' ||
                             RPAD(ROUND(((yearly_totals(i) - yearly_totals(i - 1)) / yearly_totals(i
- 1)) * 100, 2) || '%', 26) || ' |');
      ELSE
        DBMS_OUTPUT.PUT_LINE('| ' || i || '     | ' || RPAD(TO_CHAR(yearly_totals(i),
'FM999G999G999D00'), 19) || ' | ' ||
                             RPAD('N/A', 26) || ' |');
      END IF;
    END IF;
  END LOOP;

  DBMS_OUTPUT.PUT_LINE('-----------------------------------------------------------------');

  DBMS_OUTPUT.PUT_LINE(CHR(13));

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE('End of Report');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE(CHR(13));
```

```
    DBMS_OUTPUT.PUT_LINE(CHR(13));

END;
/

-- Execute the procedure
DECLARE
  v_start_year NUMBER;
  v_end_year NUMBER;
BEGIN
  -- Accept user input
  v_start_year := &start_year;
  v_end_year := &end_year;

  -- Call the procedure
  revenue_growth(v_start_year, v_end_year);
END;
/
```

Sample output:

```
~ Sales Growth Report ~
Enter value for start_year: 2014
Enter value for end_year: 2015

--------------------------------------------------------------------------------
Sales Growth Report
--------------------------------------------------------------------------------


Year: 2014
-------------------------------------------------------------------------------------
| Quarter   |   Current Sales    |   Previous Sales  |    Difference     |   Growth %  |
-------------------------------------------------------------------------------------
|   Q1      | 5,452,033.80       | -                 | -                 | -      %    |
|   Q2      | 5,489,856.95       | 5,452,033.80      | 37,823.15         | .69    %    |
|   Q3      | 5,596,077.20       | 5,489,856.95      | 106,220.25        | 1.93   %    |
|   Q4      | 5,564,750.75       | 5,596,077.20      | -31,326.45        | -.56   %    |
-------------------------------------------------------------------------------------
| Total revenue for year 2014 : 22,102,718.70                                       |
| Total revenue growth percentage for year: N/A                                     |
-------------------------------------------------------------------------------------

Year: 2015
-------------------------------------------------------------------------------------
| Quarter   |   Current Sales    |   Previous Sales  |    Difference     |   Growth %  |
-------------------------------------------------------------------------------------
|   Q1      | 5,420,402.30       | -                 | -                 | -      %    |
|   Q2      | 5,545,773.50       | 5,420,402.30      | 125,371.20        | 2.31   %    |
|   Q3      | 5,601,442.75       | 5,545,773.50      | 55,669.25         | 1.00   %    |
|   Q4      | 5,587,373.15       | 5,601,442.75      | -14,069.60        | -.25   %    |
-------------------------------------------------------------------------------------
| Total revenue for year 2015 : 22,154,991.70                                       |
| Total revenue growth percentage for year: .24%                                    |
-------------------------------------------------------------------------------------

Yearly Sales Comparison
-----------------------------------------------------------
| Year     |    Total Sales      |  Growth from Previous Year |
-----------------------------------------------------------
| 2014     | 22,102,718.70       | N/A                        |
| 2015     | 22,154,991.70       | .24%                       |
-----------------------------------------------------------


--------------------------------------------------------------------------------
End of Report
--------------------------------------------------------------------------------
```



The graph shows that in both 2014 and 2015, sales peaked in Q3 but dropped in Q4, with 2015 performing slightly better overall. Growth was highest in Q2 and Q3 for both years, with 2015 seeing stronger growth in Q2 (2.31%). However, Q4 in both years experienced a decline, indicating a consistent drop in performance at the end of the year.

From this, businesses should focus on understanding and addressing the factors leading to the Q4 decline, possibly by revising marketing or sales strategies during this period. Additionally, efforts should be made to capitalize on the growth momentum in Q2 and Q3 through targeted campaigns and operational readiness to sustain high performance during these quarters.

3.2.2 Platform Sales Query

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET FEEDBACK OFF
SET LINESIZE 180
SET PAGESIZE 180

PROMPT
PROMPT
PROMPT ~ Platform Sales Report ~

DECLARE
    start_year CHAR(4);
    end_year CHAR(4);

    CURSOR c_platform_sales IS
        WITH platform_sales AS (
            SELECT
                dd.cal_year AS year,
                dd.cal_quarter AS quarter,
                pd.PlatformName AS platform,
                SUM(o.quantity * o.unit_price) AS revenue,
                COUNT(CASE WHEN pd.PlatformName != 'In Stall' THEN o.orderID END) AS frequency --
Count only delivery orders
            FROM
                Order_Facts o
            JOIN
                Platform_dim pd ON o.Platform_key = pd.Platform_key
            JOIN
                Date_dim dd ON o.Date_key = dd.Date_key
            WHERE
                pd.PlatformName != 'In Stall'
                AND dd.cal_year BETWEEN start_year AND end_year
            GROUP BY
                dd.cal_year,
                dd.cal_quarter,
                pd.PlatformName
        ),
        yearly_sums AS (
            SELECT
                year,
                platform,
                SUM(revenue) AS total_revenue_per_year
            FROM
                platform_sales
            GROUP BY
                year, platform
        ),
        quarterly_sums AS (
```

```
            SELECT
                year,
                quarter,
                SUM(revenue) AS sum_of_revenue
            FROM
                platform_sales
            GROUP BY
                year, quarter
        )
        SELECT
            ps.year,
            ps.quarter,
            ps.platform,
            ps.revenue,
            ps.frequency, -- Include frequency in the final result
            ROUND((ps.revenue / qs.sum_of_revenue) * 100, 2) AS contribution,
            qs.sum_of_revenue,
            ys.total_revenue_per_year, -- Add yearly total revenue for platform
            ROW_NUMBER() OVER (PARTITION BY ps.year, ps.quarter ORDER BY ps.revenue DESC) AS rn,
            COUNT(*) OVER (PARTITION BY ps.year, ps.quarter) AS total_platforms
        FROM
            platform_sales ps
        JOIN
            quarterly_sums qs ON ps.year = qs.year AND ps.quarter = qs.quarter
        JOIN
            yearly_sums ys ON ps.year = ys.year AND ps.platform = ys.platform
        ORDER BY
            ps.year ASC,
            ps.quarter ASC,
            ps.revenue DESC;

    v_highest_sales VARCHAR2(100);
    v_lowest_sales VARCHAR2(100);
    v_highest_revenue NUMBER := 0;
    v_lowest_revenue NUMBER := 0;
    v_difference NUMBER := 0;
    v_yearly_highest_sales VARCHAR2(100); -- To store highest yearly platform sales
    v_yearly_lowest_sales VARCHAR2(100);  -- To store lowest yearly platform sales
    v_yearly_highest_revenue NUMBER := 0;
    v_yearly_lowest_revenue NUMBER := NULL;
    v_previous_year VARCHAR2(4);
    v_total_sales_per_year NUMBER := 0;

BEGIN
    -- Prompt for start and end year
    start_year := '&start_year';
    end_year := '&end_year';

    v_previous_year := NULL;
```

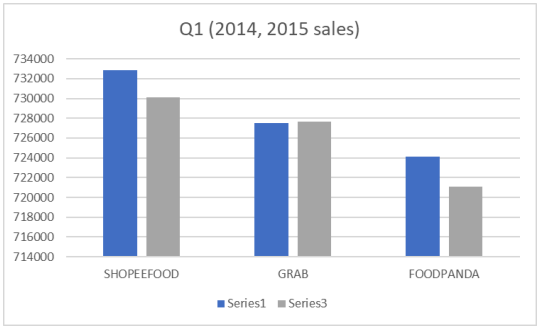
```
    v_total_sales_per_year := 0;

    DBMS_OUTPUT.PUT_LINE(CHR(13));

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE('Platform Sales Report');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE(CHR(13));

    FOR rec IN c_platform_sales LOOP
        -- New year -> Show header
        IF v_previous_year IS NULL OR rec.year != v_previous_year THEN
            IF v_previous_year IS NOT NULL THEN
                -- Display total sales and separator for the previous year

dbms_output.put_line('-----------------------------------------------------------------');
                dbms_output.put_line('| Total Sales for Year ' || v_previous_year || ' : RM ' ||
TO_CHAR(v_total_sales_per_year, '9999999.99') || RPAD(' ',20) || '|');

dbms_output.put_line('-----------------------------------------------------------------');
                dbms_output.put_line('| Highest Sales: ' || v_yearly_highest_sales || RPAD('
',22) || '|');
                dbms_output.put_line('| Lowest Sales:  ' || v_yearly_lowest_sales || RPAD(' ',22)
|| '|');
                dbms_output.put_line('| Difference: RM ' || TO_CHAR(v_yearly_highest_revenue -
v_yearly_lowest_revenue, '9999999.99') || RPAD(' ',36) || '|');

dbms_output.put_line('-----------------------------------------------------------------');
                v_total_sales_per_year := 0;  -- Reset total for the new year
            END IF;

            -- Start the new year section
            DBMS_OUTPUT.PUT_LINE(CHR(13));
            dbms_output.put_line('Year: ' || rec.year);

dbms_output.put_line('-----------------------------------------------------------------');
            dbms_output.put_line('| Quarter | Platform   |   Revenue   | Contribution | Frequency
|');

dbms_output.put_line('-----------------------------------------------------------------');

            -- Reset yearly highest and lowest values
            v_yearly_highest_revenue := 0;
            v_yearly_lowest_revenue := NULL;
            v_yearly_highest_sales := NULL;
            v_yearly_lowest_sales := NULL;
```

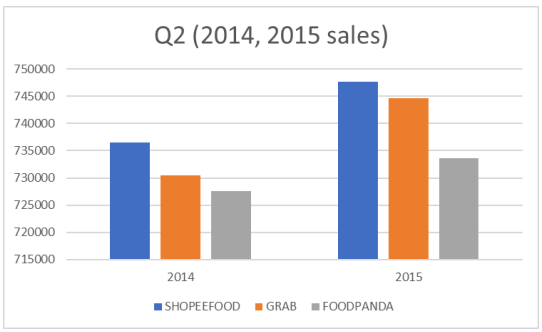
```
            v_previous_year := rec.year;
        END IF;

        -- Display the platform's data for the current quarter
        dbms_output.put_line('| ' || CASE WHEN rec.rn = 1 THEN 'Q' || rec.quarter ELSE '   ' END
||
                             '     | ' || RPAD(rec.platform, 10) || ' | ' ||
                             TO_CHAR(rec.revenue, '9999999.99') || ' | ' ||
                             RPAD(TO_CHAR(rec.contribution, '999.99') || '%', 12) || ' | ' ||
                             RPAD(TO_CHAR(rec.frequency, '999999'), 9) || ' | ');

        -- Accumulate total sales per quarter and year
        v_total_sales_per_year := v_total_sales_per_year + rec.revenue;

        -- Track yearly highest and lowest sales
        IF rec.total_revenue_per_year > v_yearly_highest_revenue THEN
            v_yearly_highest_revenue := rec.total_revenue_per_year;
            v_yearly_highest_sales := RPAD(rec.platform, 10) || ' RM ' ||
TO_CHAR(rec.total_revenue_per_year, '9999999.99');
        END IF;

        IF v_yearly_lowest_revenue IS NULL OR rec.total_revenue_per_year <
v_yearly_lowest_revenue THEN
            v_yearly_lowest_revenue := rec.total_revenue_per_year;
            v_yearly_lowest_sales := RPAD(rec.platform, 10) || ' RM ' ||
TO_CHAR(rec.total_revenue_per_year, '9999999.99');
        END IF;
    END LOOP;

    -- Display total sales for the last year after the loop ends
    IF v_previous_year IS NOT NULL THEN

dbms_output.put_line('-------------------------------------------------------------------');
        dbms_output.put_line('| Total Sales for Year ' || v_previous_year || ' : RM ' ||
TO_CHAR(v_total_sales_per_year, '9999999.99') || RPAD(' ',20) || '|');

dbms_output.put_line('-------------------------------------------------------------------');
        dbms_output.put_line('| Highest Sales: ' || v_yearly_highest_sales || RPAD(' ',22) ||
'|');
        dbms_output.put_line('| Lowest Sales:  ' || v_yearly_lowest_sales || RPAD(' ',22) ||
'|');
        dbms_output.put_line('| Difference: RM ' || TO_CHAR(v_yearly_highest_revenue -
v_yearly_lowest_revenue, '9999999.99') || RPAD(' ',36) || '|');

dbms_output.put_line('-------------------------------------------------------------------');
    END IF;


    DBMS_OUTPUT.PUT_LINE(CHR(13));
```

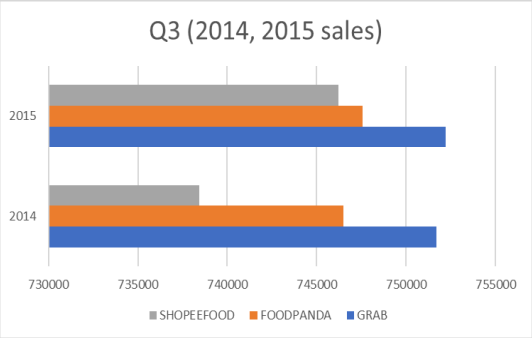
```
DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE('End of Report');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE(CHR(13));
    DBMS_OUTPUT.PUT_LINE(CHR(13));

END;
/
```

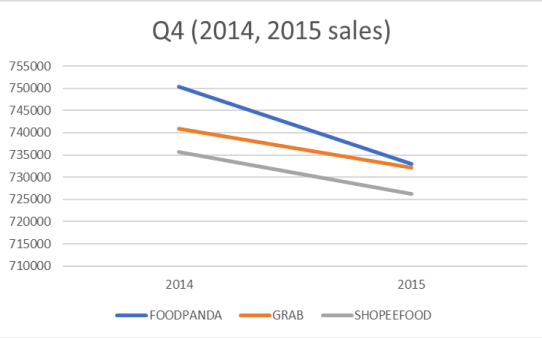
Sample output:

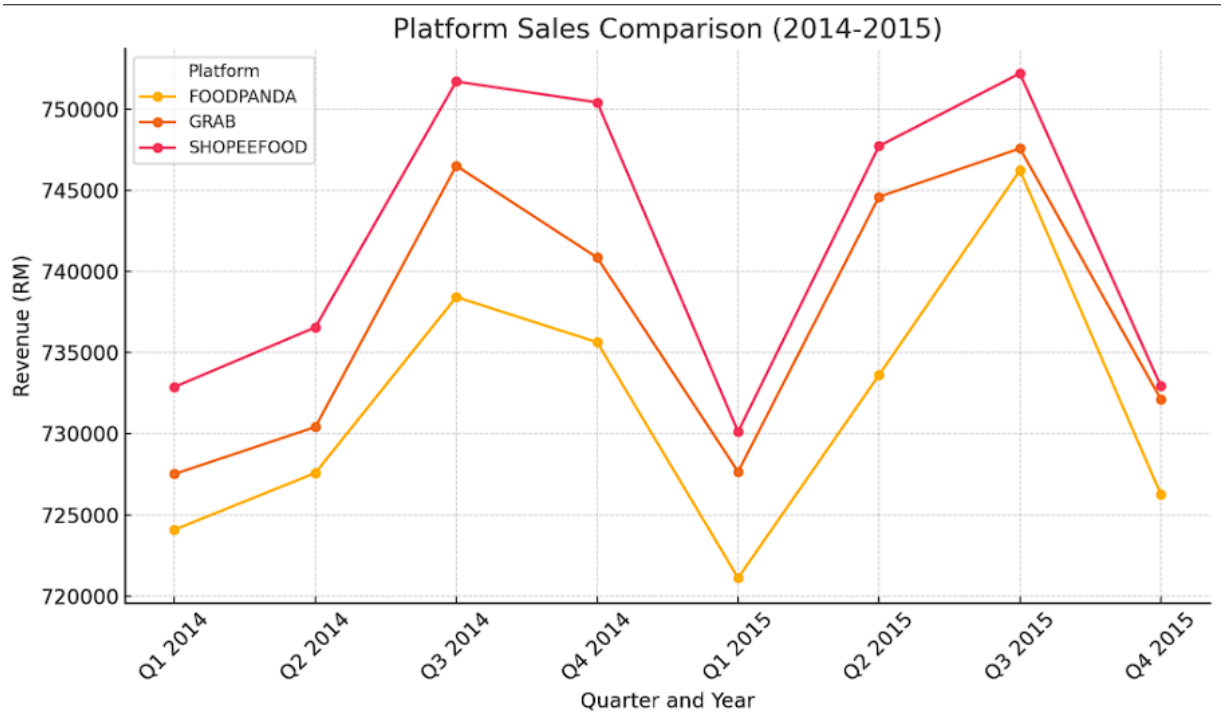
```
~ Platform Sales Report ~
Enter value for start_year: 2014
Enter value for end_year: 2015

--------------------------------------------------------------------------------
Platform Sales Report
--------------------------------------------------------------------------------


Year: 2014
-------------------------------------------------------------------
| Quarter | Platform    |   Revenue    | Contribution | Frequency |
-------------------------------------------------------------------
| Q1      | SHOPEEFOOD  |   732882.85  |   33.55%     |   17996   |
|         | GRAB        |   727519.45  |   33.30%     |   17943   |
|         | FOODPANDA   |   724086.80  |   33.15%     |   17712   |
| Q2      | SHOPEEFOOD  |   736556.75  |   33.56%     |   18081   |
|         | GRAB        |   730416.90  |   33.28%     |   17971   |
|         | FOODPANDA   |   727585.50  |   33.15%     |   18080   |
| Q3      | GRAB        |   751701.60  |   33.61%     |   18458   |
|         | FOODPANDA   |   746508.25  |   33.38%     |   18358   |
|         | SHOPEEFOOD  |   738421.65  |   33.01%     |   18091   |
| Q4      | FOODPANDA   |   750416.25  |   33.70%     |   18408   |
|         | GRAB        |   740852.05  |   33.27%     |   18151   |
|         | SHOPEEFOOD  |   735637.95  |   33.03%     |   18140   |
-------------------------------------------------------------------
| Total Sales for Year 2014 : RM  8842586.00                        |
-------------------------------------------------------------------
| Highest Sales: GRAB       RM  2950490.00                          |
| Lowest Sales:  SHOPEEFOOD RM  2943499.20                          |
| Difference: RM     6990.80                                        |
-------------------------------------------------------------------

Year: 2015
-------------------------------------------------------------------
| Quarter | Platform    |   Revenue    | Contribution | Frequency |
-------------------------------------------------------------------
| Q1      | GRAB        |   730128.45  |   33.51%     |   17922   |
|         | SHOPEEFOOD  |   727658.60  |   33.40%     |   17926   |
|         | FOODPANDA   |   721116.30  |   33.10%     |   17630   |
| Q2      | SHOPEEFOOD  |   747725.05  |   33.59%     |   18370   |
|         | GRAB        |   744601.05  |   33.45%     |   18231   |
|         | FOODPANDA   |   733614.30  |   32.96%     |   18062   |
| Q3      | GRAB        |   752201.40  |   33.49%     |   18545   |
|         | SHOPEEFOOD  |   747576.05  |   33.28%     |   18472   |
|         | FOODPANDA   |   746224.80  |   33.22%     |   18344   |
| Q4      | GRAB        |   732937.15  |   33.45%     |   17914   |
|         | FOODPANDA   |   732130.60  |   33.41%     |   18087   |
|         | SHOPEEFOOD  |   726274.90  |   33.14%     |   17859   |
-------------------------------------------------------------------
| Total Sales for Year 2015 : RM  8842188.65                        |
-------------------------------------------------------------------
| Highest Sales: GRAB       RM  2959868.05                          |
| Lowest Sales:  FOODPANDA  RM  2933086.00                          |
| Difference: RM    26782.05                                        |
-------------------------------------------------------------------


--------------------------------------------------------------------------------
End of Report
--------------------------------------------------------------------------------
```











Building on this analysis, it's clear that ShopeeFood consistently dominates the market, particularly during Q3 of both years. To capitalize on this success, ShopeeFood could invest in strategies to maintain momentum throughout the year, especially in Q4, when a sharp decline occurs. This could include holiday campaigns, loyalty programs, or expanded delivery services during this period. In contrast, Grab shows steady but less aggressive growth. A targeted strategy to push sales during peak periods like Q3 could further close the gap with ShopeeFood, especially if it focuses on enhancing user experience or introducing competitive pricing and partnerships with restaurants.

Foodpanda, however, appears to be the most vulnerable, with significant dips in both Q1 2015 and Q4 2015, suggesting structural weaknesses or external factors affecting its sales. To mitigate these losses, Foodpanda may need to re-evaluate its business model, consider investing in customer acquisition strategies, or rework its operational efficiency to handle seasonal demand fluctuations better. Collaborations with restaurants, marketing campaigns during low-performing quarters, and promotional offers can help stabilize sales and reduce volatility. Additionally, conducting market research to understand customer preferences and competitor strategies during these weaker quarters could provide valuable insights for improvement.

3.2.3 Holiday Sales Query

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET FEEDBACK OFF
SET LINESIZE 180
SET PAGESIZE 180

PROMPT
PROMPT
PROMPT ~ Holiday Sales Report ~
ACCEPT v_start_year NUMBER PROMPT 'Enter Start Year (YYYY)   : '
ACCEPT v_end_year   NUMBER PROMPT 'Enter End Year (YYYY)     : '

CREATE OR REPLACE VIEW HOLIDAY_SALES_VIEW AS
SELECT
    d.cal_year AS Year,
    d.festive_event AS Holiday,
    SUM(o.quantity * o.unit_price) AS Sales
FROM
    Order_Facts o
JOIN
    Date_dim d ON o.date_key = d.date_key
WHERE
    d.holiday_ind = 'Y'
GROUP BY
    d.cal_year, d.festive_event
ORDER BY
    d.festive_event;

CREATE OR REPLACE VIEW PUBLIC_DAY_SALES_VIEW AS
SELECT
    d.cal_year AS Year,
    SUM(o.quantity * o.unit_price) AS Sales
FROM
    Order_Facts o
JOIN
    Date_dim d ON o.date_key = d.date_key
WHERE
    d.holiday_ind = 'N'
GROUP BY
    d.cal_year;

DECLARE
  v_start_year    NUMBER := &v_start_year;
  v_end_year      NUMBER := &v_end_year;
  v_total_sales   NUMBER := 0;
  v_public_sales  NUMBER := 0;
  v_holiday_sales NUMBER := 0;
  v_holiday_count NUMBER := 0;
```

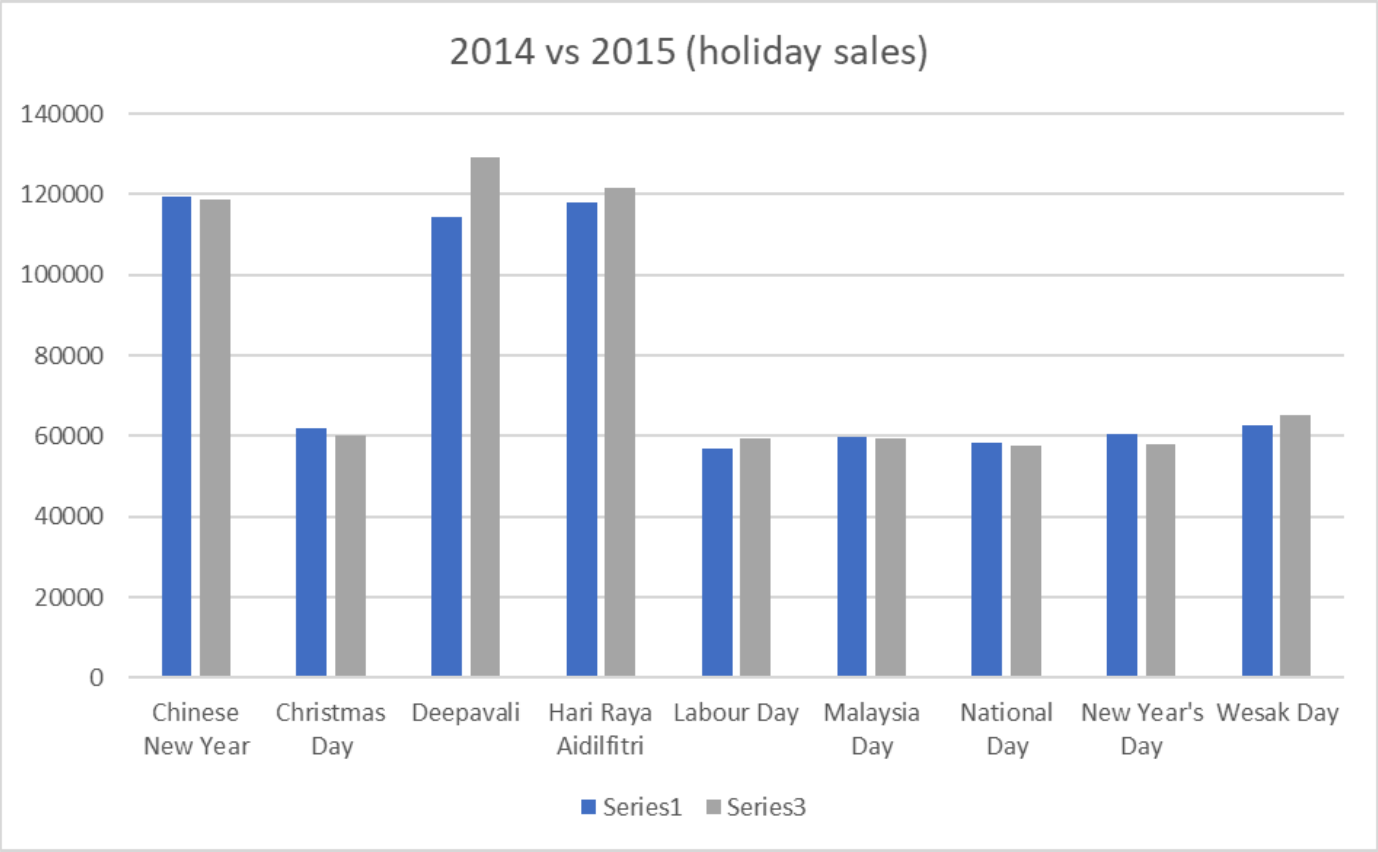
```
  v_avg_holiday   NUMBER := 0;
  v_avg_public    NUMBER := 0;

BEGIN
    DBMS_OUTPUT.PUT_LINE(CHR(13));
    DBMS_OUTPUT.PUT_LINE(CHR(13));

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE('Comparison sales for holiday and public');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE(CHR(13));

    -- Loop for each year
    FOR current_year IN v_start_year..v_end_year LOOP
        DBMS_OUTPUT.PUT_LINE(CHR(13));
        DBMS_OUTPUT.PUT_LINE(' Year : ' || current_year );

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');
        DBMS_OUTPUT.PUT_LINE('| Year | Holiday                          |   Sales (RM)   |
Contribution (%) |');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');

        -- Reset totals for the year
        v_holiday_sales := 0;
        v_public_sales  := 0;
        v_total_sales   := 0;
        v_holiday_count := 0;

        -- Fetch total holiday sales for the year (for calculating contribution percentage)
        SELECT
            SUM(Sales)
        INTO
            v_holiday_sales
        FROM
            HOLIDAY_SALES_VIEW
        WHERE
            Year = current_year;

        -- Fetch and display holiday sales for each holiday from the view
        FOR holiday_sales_rec IN (
            SELECT
                Year, Holiday, Sales
            FROM
```

```
            HOLIDAY_SALES_VIEW
        WHERE
            Year = current_year
    ) LOOP
        -- Calculate contribution percentage for each holiday
        DECLARE
            v_contribution NUMBER := (holiday_sales_rec.Sales / v_holiday_sales) * 100;
        BEGIN
            -- Display each holiday sales with contribution percentage
            DBMS_OUTPUT.PUT_LINE(
                ' | ' || RPAD(holiday_sales_rec.Year, 5) ||
                ' | ' || RPAD(SUBSTR(holiday_sales_rec.Holiday, 1, 30), 30) ||
                ' | ' || TO_CHAR(holiday_sales_rec.Sales, '99,999,999.99') ||
                ' | ' || TO_CHAR(v_contribution, '999.99') || '%' || RPAD(' ',8) ||' |'
            );
        END;

        -- Accumulate holiday count
        v_holiday_count := v_holiday_count + 1;
    END LOOP;

    -- Fetch public day sales for the current year from the view
    SELECT
        Sales
    INTO
        v_public_sales
    FROM
        PUBLIC_DAY_SALES_VIEW
    WHERE
        Year = current_year;

    -- Calculate total sales for the year
    v_total_sales := v_holiday_sales + v_public_sales;
    v_avg_holiday := (v_holiday_sales / v_holiday_count);
    v_avg_public  := (v_public_sales / (365 - v_holiday_count));

    -- Display the total sales and public day sales

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE(' | Total Holiday Sales:                    |'||
TO_CHAR(v_holiday_sales, '999,999,999.99') || RPAD(' ',19) || ' |');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE(' | Public Day Sales:                       |' ||
TO_CHAR(v_public_sales, '999,999,999.99') || RPAD(' ',19) || ' |');
```

```
DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');

        -- Display the average holiday sales and public day sales
        IF v_holiday_count > 0 THEN
            DBMS_OUTPUT.PUT_LINE(' | Average Holiday Sales:                | ' ||
TO_CHAR(v_avg_holiday, '99,999,999.99') || RPAD(' ',19) || ' |');
        ELSE
            DBMS_OUTPUT.PUT_LINE(' | Average Holiday Sales:                | 0.00 |');
        END IF;

        DBMS_OUTPUT.PUT_LINE(' | Average Public Day Sales:             | ' ||
TO_CHAR(v_avg_public, '99,999,999.99') || RPAD(' ',19) || ' |');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');

        -- Calculate and display the difference
        DBMS_OUTPUT.PUT_LINE(' | Difference In Avg (Holiday vs Public): |' ||
TO_CHAR((v_avg_holiday - v_avg_public), '999,999,999.99') || RPAD(' ',19) || ' |');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');

    END LOOP;

    DBMS_OUTPUT.PUT_LINE(CHR(13));

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE('End of Report');

DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------
---');
    DBMS_OUTPUT.PUT_LINE(CHR(13));
    DBMS_OUTPUT.PUT_LINE(CHR(13));

END;
/
```

Sample output:

```
~ Holiday Sales Report ~
Enter Start Year (YYYY)   : 2014
Enter End Year (YYYY)     : 2015


--------------------------------------------------------------------------------
Comparison sales for holiday and public
--------------------------------------------------------------------------------


Year : 2014
--------------------------------------------------------------------------------
| Year | Holiday                        |   Sales (RM)   | Contribution (%) |
--------------------------------------------------------------------------------
| 2014  | Chinese New Year              |     119,303.35 |   16.76%         |
| 2014  | Christmas Day                 |      62,015.30 |    8.71%         |
| 2014  | Deepavali                     |     114,462.05 |   16.08%         |
| 2014  | Hari Raya Aidilfitri          |     117,898.35 |   16.56%         |
| 2014  | Labour Day                    |      56,910.95 |    7.99%         |
| 2014  | Malaysia Day                  |      59,813.95 |    8.40%         |
| 2014  | National Day                  |      58,309.60 |    8.19%         |
| 2014  | New Year's Day                |      60,529.55 |    8.50%         |
| 2014  | Wesak Day                     |      62,757.80 |    8.81%         |
--------------------------------------------------------------------------------
| Total Holiday Sales:                |     712,000.90                        |
--------------------------------------------------------------------------------
| Public Day Sales:                   |  21,390,717.80                        |
--------------------------------------------------------------------------------
| Average Holiday Sales:              |      79,111.21                        |
| Average Public Day Sales:           |      60,086.29                        |
--------------------------------------------------------------------------------
| Difference In Avg (Holiday vs Public): |      19,024.93                     |
--------------------------------------------------------------------------------

Year : 2015
--------------------------------------------------------------------------------
| Year | Holiday                        |   Sales (RM)   | Contribution (%) |
--------------------------------------------------------------------------------
| 2015  | Chinese New Year              |     118,500.10 |   16.25%         |
| 2015  | Christmas Day                 |      60,241.80 |    8.26%         |
| 2015  | Deepavali                     |     129,037.30 |   17.70%         |
| 2015  | Hari Raya Aidilfitri          |     121,627.40 |   16.68%         |
| 2015  | Labour Day                    |      59,539.25 |    8.17%         |
| 2015  | Malaysia Day                  |      59,435.05 |    8.15%         |
| 2015  | National Day                  |      57,475.50 |    7.88%         |
| 2015  | New Year's Day                |      58,120.75 |    7.97%         |
| 2015  | Wesak Day                     |      65,085.50 |    8.93%         |
--------------------------------------------------------------------------------
| Total Holiday Sales:                |     729,062.65                        |
--------------------------------------------------------------------------------
| Public Day Sales:                   |  21,425,929.05                        |
--------------------------------------------------------------------------------
| Average Holiday Sales:              |      81,006.96                        |
| Average Public Day Sales:           |      60,185.19                        |
--------------------------------------------------------------------------------
| Difference In Avg (Holiday vs Public): |      20,821.77                     |
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
End of Report
--------------------------------------------------------------------------------
```



he bar chart compares holiday sales for 2014 (Series 1) and 2015 (Series 3) across major holidays. For most holidays, 2015 outperformed 2014, particularly during Hari Raya Aidilfitri and Chinese New Year, which show significant increases in sales. However, some holidays like Christmas Day saw nearly equal sales in both years, and Labour Day shows relatively lower sales in both years compared to other holidays. This suggests that holidays like Deepavali and Chinese New Year consistently drive higher revenue, with Hari Raya showing the most significant improvement between the two years.

From this, businesses can focus their marketing and promotional efforts on the holidays with historically higher sales potential, such as Hari Raya Aidilfitri, Deepavali, and Chinese New Year, to maximize revenue. Additionally, the data suggests there may be room to grow during Labour Day and Malaysia Day, where sales were relatively low. Targeted strategies such as holiday-specific promotions or partnerships with local events could help boost performance during these underperforming periods.

3.3 Tong Chun Mun

3.3.1 Total Sales and Orders comparison for each menu item type in a specific quarter between 2 specific years

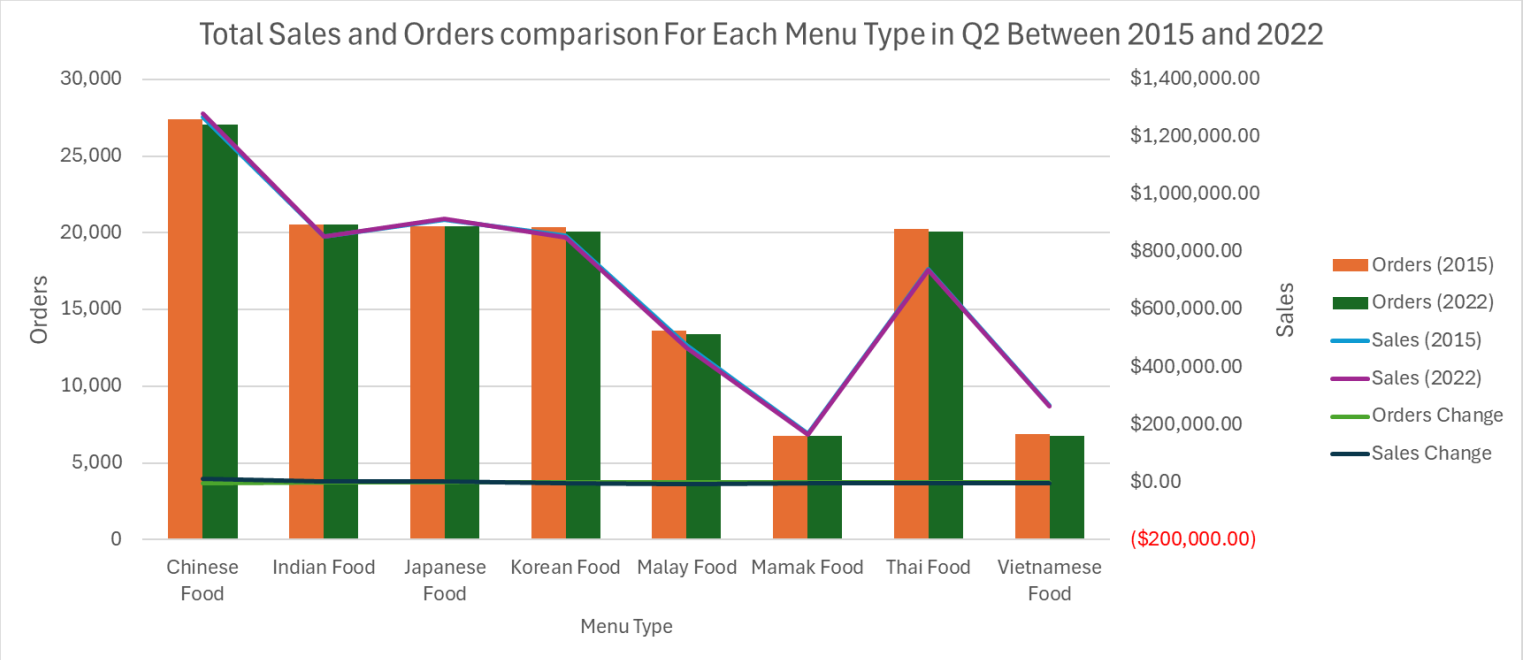
```
SET linesize 150
SET pagesize 120
ALTER SESSION SET NLS_DATE_FORMAT = 'dd-MON-YYYY';

-- Prompt user for input
ACCEPT year1 CHAR FORMAT A4 PROMPT ' Enter first year (2014-2023): '
ACCEPT year2 CHAR FORMAT A4 PROMPT ' Enter second year (2014-2023): '
ACCEPT quarter_number CHAR FORMAT A1 PROMPT ' Enter quarter (1-4): '

-- Set column formats and headings
COLUMN menu_type FORMAT A20 HEADING 'Menu Type'
COLUMN year1_sales FORMAT $999,999,999.99 HEADING '(&year1)|Sales'
COLUMN year2_sales FORMAT $999,999,999.99 HEADING '(&year2)|Sales'
COLUMN sales_diff FORMAT $999,999,999.99 HEADING 'Sales Diff'
COLUMN pct_change FORMAT 990.99 HEADING 'Change(%)'
COLUMN year1_orders FORMAT 999,999 HEADING '(&year1)|Orders'
COLUMN year2_orders FORMAT 999,999 HEADING '(&year2)|Orders'
COLUMN orders_diff FORMAT 999,999 HEADING 'Orders Diff'

-- Title for the report
TTITLE -
CENTER 'Total Sales and Orders Comparison For Each Menu Type' -
SKIP 1 -
CENTER 'Between Q&quarter_number Of Year &year1 and &year2' -
RIGHT 'Date: ' _DATE -
SKIP 1 RIGHT 'Page ' -
FORMAT 999 SQL.PNO -
SKIP 2

-- Break on total row for yearly aggregates
BREAK ON REPORT SKIP 1

-- Compute totals for relevant columns
COMPUTE SUM LABEL 'TOTAL:' OF year1_sales year2_sales sales_diff year1_orders year2_orders
orders_diff ON REPORT

WITH Aggregated_Quarterly_Data AS (
  SELECT
    B.stalltype AS menu_type,
    D.cal_year,
    D.cal_quarter,
    SUM(O.unit_price * O.quantity) AS total_sales,
    COUNT(O.orderid) AS total_orders
```

```
  FROM
    Order_Facts O
  JOIN
    Menu_dim M ON O.menu_key = M.menu_key
  JOIN
    Date_dim D ON O.date_key = D.date_key
  JOIN
    Business_dim B ON O.business_key = B.business_key
  WHERE
    D.cal_quarter = &quarter_number
    AND (D.cal_year = &year1 OR D.cal_year = &year2)
  GROUP BY
    B.stalltype, D.cal_year, D.cal_quarter
),
Yearly_Totals AS (
  SELECT
    menu_type,
    SUM(CASE WHEN cal_year = &year1 THEN total_sales ELSE 0 END) AS year1_sales,
    SUM(CASE WHEN cal_year = &year2 THEN total_sales ELSE 0 END) AS year2_sales,
    SUM(CASE WHEN cal_year = &year1 THEN total_orders ELSE 0 END) AS year1_orders,
    SUM(CASE WHEN cal_year = &year2 THEN total_orders ELSE 0 END) AS year2_orders
  FROM
    Aggregated_Quarterly_Data
  GROUP BY
    menu_type
),
Comparison AS (
  SELECT
    menu_type,
    year1_sales,
    year2_sales,
    year1_orders,
    year2_orders,
    year2_sales - year1_sales AS sales_diff,
    ((year2_sales - year1_sales) / NULLIF(year1_sales, 0)) * 100 AS pct_change,
    year2_orders - year1_orders AS orders_diff
  FROM
    Yearly_Totals
)
SELECT
  menu_type,
  year1_orders,
  year2_orders,
  orders_diff,
  year1_sales,
  year2_sales,
  sales_diff,
  pct_change
FROM
```

```
  Comparison
UNION ALL
SELECT
  'TOTAL:' AS menu_type,
  SUM(year1_orders) AS year1_orders,
  SUM(year2_orders) AS year2_orders,
  SUM(orders_diff) AS orders_diff,
  SUM(year1_sales) AS year1_sales,
  SUM(year2_sales) AS year2_sales,
  SUM(sales_diff) AS sales_diff,
  (CASE
    WHEN SUM(year1_sales) = 0 THEN NULL
    ELSE (SUM(year2_sales) - SUM(year1_sales)) / SUM(year1_sales) * 100
  END) AS pct_change
FROM
  Comparison
ORDER BY
  menu_type;

-- Clear computes and columns after the query
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF
```

Output:

```
Session altered.

 Enter first year (2014-2023): 2015
 Enter second year (2014-2023): 2022
 Enter quarter (1-4): 2
old  17:     D.cal_quarter = &quarter_number
new  17:     D.cal_quarter = 2
old  18:     AND (D.cal_year = &year1 OR D.cal_year = &year2)
new  18:     AND (D.cal_year = 2015 OR D.cal_year = 2022)
old  25:     SUM(CASE WHEN cal_year = &year1 THEN total_sales ELSE 0 END) AS year1_sales,
new  25:     SUM(CASE WHEN cal_year = 2015 THEN total_sales ELSE 0 END) AS year1_sales,
old  26:     SUM(CASE WHEN cal_year = &year2 THEN total_sales ELSE 0 END) AS year2_sales,
new  26:     SUM(CASE WHEN cal_year = 2022 THEN total_sales ELSE 0 END) AS year2_sales,
old  27:     SUM(CASE WHEN cal_year = &year1 THEN total_orders ELSE 0 END) AS year1_orders,
new  27:     SUM(CASE WHEN cal_year = 2015 THEN total_orders ELSE 0 END) AS year1_orders,
old  28:     SUM(CASE WHEN cal_year = &year2 THEN total_orders ELSE 0 END) AS year2_orders
new  28:     SUM(CASE WHEN cal_year = 2022 THEN total_orders ELSE 0 END) AS year2_orders

                                    Total Sales and Orders Comparison For Each Menu Type
                                            Between Q2 Of Year 2015 and 2022                                Date: 15-SEP-2024
                                                                                                                 Page    1

                    (2015)   (2022)                    (2015)          (2022)
Menu Type           Orders   Orders Orders Diff          Sales           Sales       Sales Diff Change(%)
------------------- -------- -------- ----------- --------------- --------------- -------------- ---------
Chinese Food         27,374   27,079        -295    $1,269,581.75   $1,280,497.20    $10,915.45      0.86
Indian Food          20,516   20,541          25      $852,459.70     $854,620.20     $2,160.50      0.25
Japanese Food        20,441   20,430         -11      $913,013.95     $914,300.25     $1,286.30      0.14
Korean Food          20,367   20,098        -269      $855,582.25     $850,631.75    -$4,950.50     -0.58
Malay Food           13,627   13,412        -215      $475,708.25     $467,165.50    -$8,542.75     -1.80
Mamak Food            6,775    6,738         -37      $168,167.00     $164,001.50    -$4,165.50     -2.48
TOTAL:              136,245  135,157      -1,088    $5,542,093.95   $5,531,085.65   -$11,008.30     -0.20
Thai Food            20,257   20,080        -177      $740,104.05     $737,581.50    -$2,522.55     -0.34
Vietnamese Food       6,888    6,779        -109      $267,477.00     $262,287.75    -$5,189.25     -1.94
                    -------- -------- ----------- --------------- --------------- --------------
TOTAL:               272,490  270,314      -2,176   $11,084,187.90   $11,062,171.30   -$22,016.60


9 rows selected.
```



From the report and chart, we can see that overall sales and orders slightly declined between 2015 and 2022, with a total decrease of 2,176 orders and $22,016.60 in sales. Some menu types like Chinese and Indian food showed minor growth in sales, while others, such as Malay and Mamak food, experienced a decline. This suggests that businesses should focus on understanding customer preferences to improve or maintain sales in declining categories.

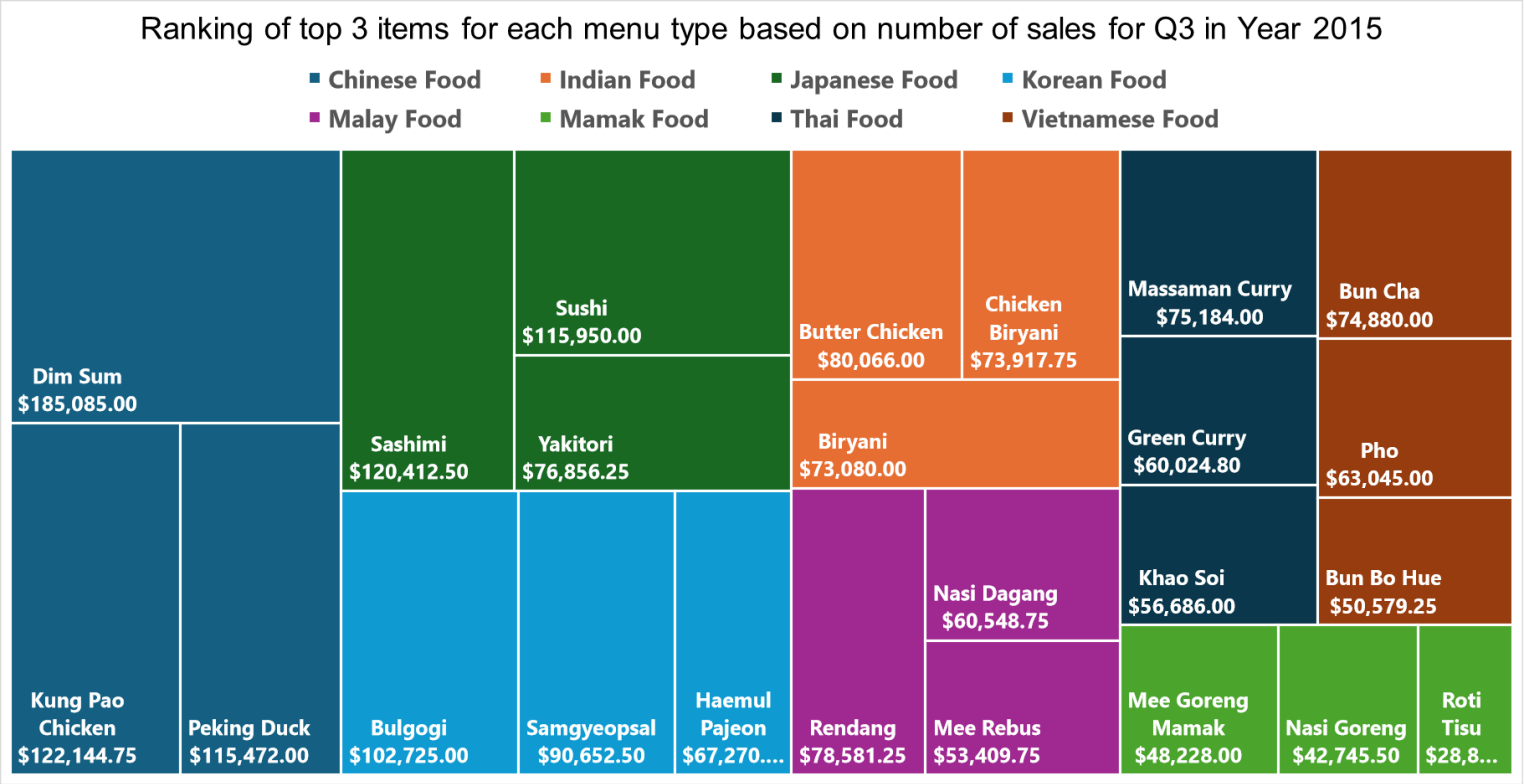
3.3.2 Ranking of top 3 items for each menu type based on number of sales for a specific quarter of a specific year

```
-- Set environment settings for the report
SET linesize 99
SET pagesize 150
alter session set nls_date_format = 'dd/mm/yyyy';
ACCEPT TargetYear DATE FORMAT 'yyyy' PROMPT 'Enter a year (YYYY): ';
ACCEPT TargetQuarter CHAR FORMAT A02 PROMPT 'Enter a quarter (1/2/3/4): ';
COLUMN Category FORMAT A20 HEADING 'Category';
COLUMN menuitem FORMAT A45 HEADING 'Menu Item';
COLUMN Rank FORMAT 99 HEADING "Rank";
COLUMN TotalSales_EachProduct FORMAT $999,999,999.99 HEADING 'Total Revenue';
COLUMN Percentage_Over_Total_Revenue FORMAT A10 HEADING '( % )';
TTITLE CENTER 'Ranking Of Top 3 Items for Each Menu Type based on Number Of Sales (%)' -
SKIP 1 CENTER 'For Each Category within Q'&TargetQuarter' in Year '&TargetYear'' -
SKIP 1 RIGHT 'Date: ' _DATE SKIP 1 RIGHT 'Page ' FORMAT 999 SQL.PNO SKIP 2
BREAK ON Category SKIP 2
COMPUTE SUM LABEL 'Total : ' OF TotalSales_EachProduct ON Category

-- Create or replace the view for ranking each Menu Item
CREATE OR REPLACE VIEW RankingEachmenuitem AS
WITH calTotalSales AS (
    SELECT
        cal_quarter AS Cal_Quarter,
        cal_year AS Cal_Year,
        SUM(unit_price * quantity) AS TotalSales
    FROM
        Order_Facts O
        JOIN Date_dim DD ON O.DATE_KEY = DD.DATE_KEY
    WHERE
        cal_quarter = '&TargetQuarter'
        AND cal_year = '&TargetYear'
    GROUP BY
        cal_quarter, cal_year
),
calTotalSales_EachProduct AS (
    SELECT
        DD.cal_quarter AS Cal_Quarter,
        DD.cal_year AS Cal_Year,
        B.stallType AS Category,
        M.FOODNDRINK AS menuitem,
        SUM(O.unit_price * O.quantity) AS TotalSales_EachProduct
    FROM
        Order_Facts O
        JOIN Date_dim DD ON O.DATE_KEY = DD.DATE_KEY
        JOIN Menu_dim M ON O.Menu_key = M.MENU_KEY
        JOIN business_dim B ON O.Business_key = B.business_key
    WHERE
```

```
        DD.cal_quarter = '&TargetQuarter'
        AND DD.cal_year = '&TargetYear'
    GROUP BY
        DD.cal_quarter, DD.cal_year, B.stallType, M.FOODNDRINK
),
resultTable AS (
    SELECT
        Category,
        menuitem,
        DENSE_RANK() OVER (PARTITION BY Category ORDER BY TotalSales_EachProduct DESC) AS Rank,
        TotalSales_EachProduct,
        TO_CHAR(TotalSales_EachProduct / TotalSales * 100, '90.99') || '%' AS
Percentage_Over_Total_Revenue
    FROM
        calTotalSales
        JOIN calTotalSales_EachProduct ON calTotalSales.Cal_Quarter =
calTotalSales_EachProduct.Cal_Quarter
        AND calTotalSales.Cal_Year = calTotalSales_EachProduct.Cal_Year
)
SELECT
    Category,
    menuitem,
    Rank,
    TotalSales_EachProduct,
    Percentage_Over_Total_Revenue
FROM
    resultTable
WHERE
    Rank <= 3
WITH READ ONLY CONSTRAINT read_only_Rankingmenuitem;

-- Display the report
SELECT * FROM RankingEachmenuitem;
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF
```

Output:

```
Session altered.

Enter a year (YYYY): 2015
Enter a quarter (1/2/3/4): 3
old  11:         cal_quarter = '&TargetQuarter'
new  11:         cal_quarter = '3'
old  12:         AND cal_year = '&TargetYear'
new  12:         AND cal_year = '2015'
old  29:         DD.cal_quarter = '&TargetQuarter'
new  29:         DD.cal_quarter = '3'
old  30:         AND DD.cal_year = '&TargetYear'
new  30:         AND DD.cal_year = '2015'

View created.


               Ranking Of Top 3 Items for Each Menu Type based on Number Of Sales (%)
                              For Each Category within Q3 in Year 2015
                                                                                Date: 15/09/202
                                                                                       Page

Category             Menu Item                                      Rank    Total Revenue ( % )
-------------------- ---------------------------------------------- ---- ---------------- ---------
Chinese Food         Dim Sum                                           1      $185,085.00   3.33%
                     Kung Pao Chicken                                  2      $122,144.75   2.20%
                     Peking Duck                                       3      $115,472.00   2.08%
********************                                                     ----------------
Total :                                                                      $422,701.75


Indian Food          Butter Chicken                                    1       $80,066.00   1.44%
                     Chicken Biryani                                   2       $73,917.75   1.33%
                     Biryani                                           3       $73,080.00   1.31%
********************                                                     ----------------
Total :                                                                      $227,063.75


Japanese Food        Sashimi                                           1      $120,412.50   2.17%
                     Sushi                                             2      $115,950.00   2.09%
                     Yakitori                                          3       $76,856.25   1.38%
********************                                                     ----------------
Total :                                                                      $313,218.75


Korean Food          Bulgogi                                           1      $102,725.00   1.85%
                     Samgyeopsal                                       2       $90,652.50   1.63%
                     Haemul Pajeon                                     3       $67,270.50   1.21%
********************                                                     ----------------
Total :                                                                      $260,648.00


Malay Food           Rendang                                           1       $78,581.25   1.41%
                     Nasi Dagang                                       2       $60,548.75   1.09%
                     Mee Rebus                                         3       $53,409.75   0.96%
********************                                                     ----------------
Total :                                                                      $192,539.75


Mamak Food           Mee Goreng Mamak                                  1       $48,228.00   0.87%
                     Nasi Goreng                                       2       $42,745.50   0.77%
                     Roti Tisu                                         3       $28,868.00   0.52%
********************                                                     ----------------
Total :                                                                      $119,841.50


Thai Food            Massaman Curry                                    1       $75,184.00   1.35%
                     Green Curry                                       2       $60,024.80   1.08%
                     Khao Soi                                          3       $56,686.00   1.02%
********************                                                     ----------------
Total :                                                                      $191,894.80


Vietnamese Food      Bun Cha                                           1       $74,880.00   1.35%
                     Pho                                               2       $63,045.00   1.13%
                     Bun Bo Hue                                        3       $50,579.25   0.91%
********************                                                     ----------------
Total :                                                                      $188,504.25
```



Based on the report and the treemap above, we can analyze the top 3 items for each menu type based on the number of sales for Q3 in 2015. We can make strategic decisions such as prioritizing inventory for high-performing items, tailoring promotions to boost sales of top sellers, and adjusting marketing efforts to highlight these popular items. Additionally, understanding which items are leading in sales allows for targeted adjustments in menu types, potentially improving overall profitability and customer satisfaction. For example, we can increase the inventory of Dim Sum as it has the highest sales, maximizing the profits.

3.3.3 Total sales and sales changes of a specific menu item of a specific stall from 2014 to 2023

```
SET linesize 120
SET pagesize 120
ALTER SESSION SET NLS_DATE_FORMAT = 'dd-MON-YYYY';


ACCEPT In_StallID CHAR FORMAT 'A4' PROMPT ' Enter a Stall ID (S101-S120): '

-- Retrieve valid menu options for the selected Stall
COLUMN menuID FORMAT A4 HEADING "Menu ID";
COLUMN FOODNDRINK FORMAT A40 HEADING "Item";

SELECT DISTINCT
    M.menuID,
    M.FOODNDRINK
FROM
    Order_Facts O
JOIN
```

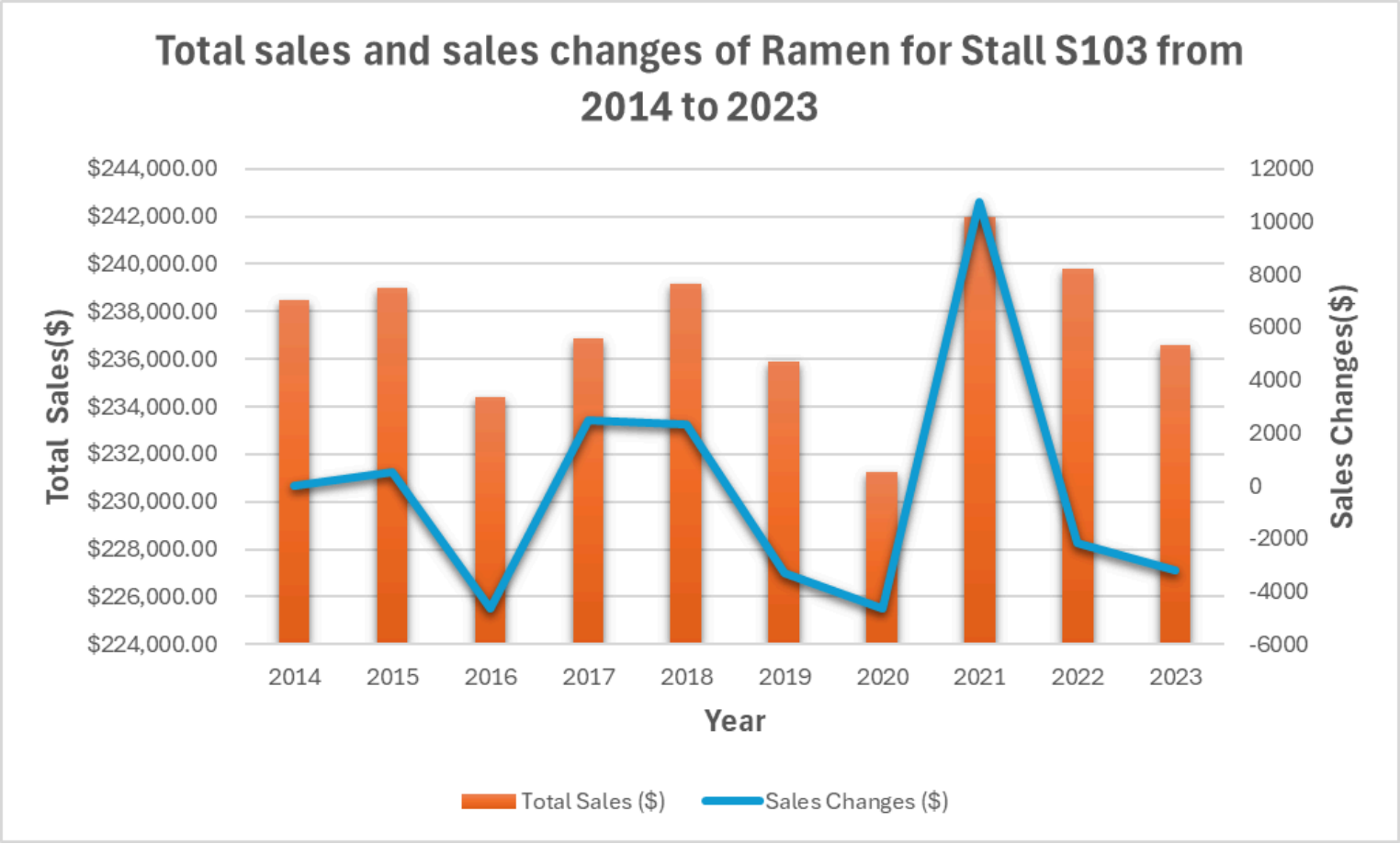
```
    Menu_dim M ON O.menu_key = M.menu_key
JOIN
    Business_dim B ON O.business_key = B.business_key
WHERE
    B.stallID = '&In_StallID';

ACCEPT In_MenuID CHAR FORMAT 'A4' PROMPT ' Enter a Menu Item Code (MenuID): '

-- Setting up columns for the final report
COLUMN MenuItem FORMAT A25 HEADING "Menu Item";
COLUMN cal_Year FORMAT 9999 HEADING "Year";
COLUMN TotalSales FORMAT 99,999,990.99 HEADING "Total Sales ($)";
COLUMN SalesPercentage FORMAT 90.99 HEADING "Sales Percentage |By Menu Item (%)";
COLUMN SalesChanges FORMAT 99,999,990.99 HEADING "Sales Changes ($)";
COLUMN ChangesPercentage FORMAT 9990.99 HEADING "Changes Percentage (%)";
COLUMN Ranking FORMAT 99 HEADING "Ranking";

-- Title setup
TTITLE -
CENTER 'Sales and Sales Changes OF Menu Item &In_MenuID ' -
SKIP 1 -
CENTER 'for Stall &In_StallID from 2014 to 2023' -
RIGHT 'Date: ' _DATE -
SKIP 1 RIGHT 'Page ' -
FORMAT 999 SQL.PNO -
SKIP 2

-- Adding BREAK command to group and summarize data
BREAK ON stallID ON MenuItem

COMPUTE SUM LABEL 'Total Sales: ' OF TotalSales SalesChanges ON MenuItem
COMPUTE SUM OF SalesChanges ON MenuItem

-- Main Query
CREATE OR REPLACE VIEW YearlyStallSalesView AS
SELECT
    D.Cal_Year,
    B.stallID,
    M.menuID,
    M.FoodNDrink,
    SUM(O.unit_price * O.quantity) AS TotalSales
FROM
    Order_Facts O
JOIN
    Date_dim D ON O.date_key = D.date_key
JOIN
    Business_dim B ON O.business_key = B.business_key
JOIN
    Menu_dim M ON O.menu_key = M.menu_key
```

```
WHERE
    B.stallID = '&In_StallID'
    AND M.menuID = '&In_MenuID'
GROUP BY
    D.Cal_Year, B.stallID, M.menuID, M.FoodNDrink;

CREATE OR REPLACE VIEW StallTotalYearlySalesView AS
SELECT
    D.Cal_Year,
    B.stallID,
    SUM(O.unit_price * O.quantity) AS totalStallSales
FROM
    Order_Facts O
JOIN
    Date_dim D ON O.date_key = D.date_key
JOIN
    Business_dim B ON O.business_key = B.business_key
WHERE
    B.stallID = '&In_StallID'
GROUP BY
    D.Cal_Year, B.stallID;

CREATE OR REPLACE VIEW YearlySalesChangeView AS
SELECT
    YSSV.Cal_Year,
    YSSV.stallID,
    YSSV.menuID,
    YSSV.TotalSales,
    YSSV.TotalSales - LAG(YSSV.TotalSales) OVER (PARTITION BY YSSV.stallID, YSSV.menuID ORDER BY
YSSV.Cal_Year) AS SalesChanges
FROM
    YearlyStallSalesView YSSV;

CREATE OR REPLACE VIEW YearlySalesRankingView AS
SELECT
    YSSV.Cal_Year,
    YSSV.stallID,
    YSSV.menuID,
    YSSV.TotalSales,
    RANK() OVER (PARTITION BY YSSV.stallID ORDER BY YSSV.TotalSales DESC) AS Ranking
FROM
    YearlyStallSalesView YSSV;

CREATE OR REPLACE VIEW FinStallMenuPerformView AS
SELECT
    YSSV.FoodNDrink AS MenuItem,
    YSSV.Cal_Year AS Year,
    YSSV.TotalSales AS TotalSales,
    ROUND(YSSV.TotalSales / STYSV.totalStallSales * 100, 2) AS SalesPercentage,
```

```
    YSCV.SalesChanges AS SalesChanges,
    ROUND(YSCV.SalesChanges / LAG(YSSV.TotalSales) OVER (PARTITION BY YSSV.stallID, YSSV.menuID
ORDER BY YSSV.Cal_Year) * 100, 2) AS ChangesPercentage,
    YSRV.Ranking
FROM
    YearlyStallSalesView YSSV
JOIN
    StallTotalYearlySalesView STYSV
    ON YSSV.Cal_Year = STYSV.Cal_Year AND YSSV.stallID = STYSV.stallID
JOIN
    YearlySalesChangeView YSCV
    ON YSSV.Cal_Year = YSCV.Cal_Year AND YSSV.stallID = YSCV.stallID AND YSSV.menuID =
YSCV.menuID
JOIN
    YearlySalesRankingView YSRV
    ON YSSV.Cal_Year = YSRV.Cal_Year AND YSSV.stallID = YSRV.stallID AND YSSV.menuID =
YSRV.menuID
ORDER BY YSSV.Cal_Year;

SELECT * FROM FinStallMenuPerformView;

-- Clear formats, titles, and configurations
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF
```

Output:

```
Session altered.

 Enter a Stall ID (S101-S120): S103
old  11:     B.stallID = '&In_StallID'
new  11:     B.stallID = 'S103'

Menu Item
---- ---------------------------------------
M014 Sashimi
M013 Tempura
M015 Yakitori
M011 Sushi
M012 Ramen

 Enter a Menu Item Code (MenuID): M012
old  17:     B.stallID = '&In_StallID'
new  17:     B.stallID = 'S103'
old  18:     AND M.menuID = '&In_MenuID'
new  18:     AND M.menuID = 'M012'

View created.

old  13:     B.stallID = '&In_StallID'
new  13:     B.stallID = 'S103'

View created.


View created.


View created.


View created.


                                    Sales and Sales Changes OF Menu Item M012
                                         for Stall S103 from 2014 to 2023                          Date: 15-SEP-2024
                                                                                                          Page    1

                                                  Sales Percentage
Menu Item                      YEAR Total Sales ($)  By Menu Item (%) Sales Changes ($) Changes Percentage (%) Ranking
------------------------ ---------- -------------- ---------------- ---------------- ---------------------- -------
Ramen                          2014      238,481.50            13.74                                              5
                               2015      239,018.00            13.56           536.50                   0.22       4
                               2016      234,378.00            13.15        -4,640.00                  -1.94       9
                               2017      236,872.00            13.26         2,494.00                   1.06       6
                               2018      239,177.50            13.64         2,305.50                   0.97       3
                               2019      235,886.00            13.39        -3,291.50                  -1.38       8
                               2020      231,246.00            13.20        -4,640.00                  -1.97      10
                               2021      241,976.00            13.79        10,730.00                   4.64       1
                               2022      239,801.00            13.63        -2,175.00                  -0.90       2
                               2023      236,596.50            13.31        -3,204.50                  -1.34       7
************************             --------------                 ----------------
Total Sales:                          2,373,432.50                        -1,885.00

10 rows selected.
```



The report and the chart shows the total sales and sales changes of Ramen for Stall S103 from 2014 to 2023. Analyzing sales changes and percentage changes over time can reveal trends and patterns, such as whether a menu item is gaining or losing popularity. This analysis helps in optimizing inventory, targeting marketing efforts, and making strategic adjustments to enhance sales and profitability.

3.4 Terence Tiu Chuan Jie

3.3.1 Revenue and Customer Breakdown by each menu item and Gender for a specific quarter Year and Quarter with in the rank of top revenue choose

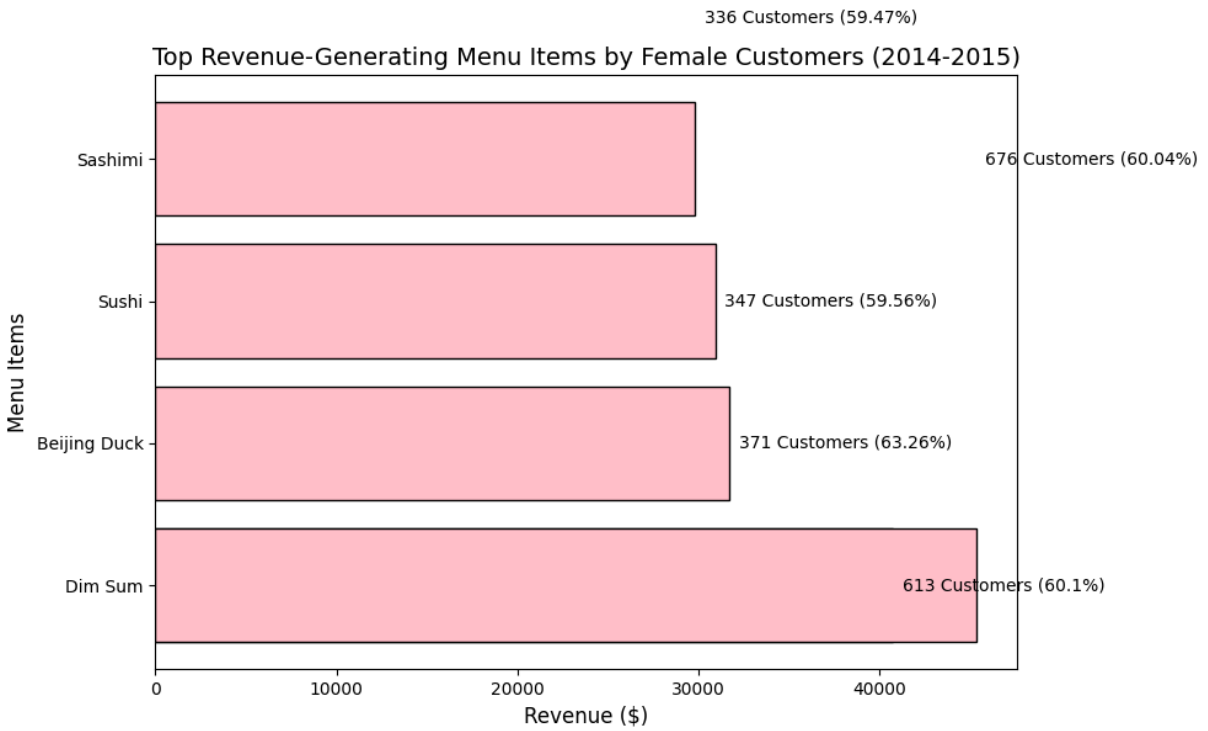
```
SET SERVEROUTPUT ON
CLEAR SCREEN
SET linesize 150
SET pagesize 190
ALTER SESSION SET nls_date_format = 'dd/mm/yyyy';

-- Accept user input for start year, end year, quarter, and rank
ACCEPT TargetYearS CHAR FORMAT 'A4' PROMPT "Enter a start year (YYYY): " DEFAULT '2014';
ACCEPT TargetYearE CHAR FORMAT 'A4' PROMPT "Enter an end year (YYYY): " DEFAULT '2015';
```

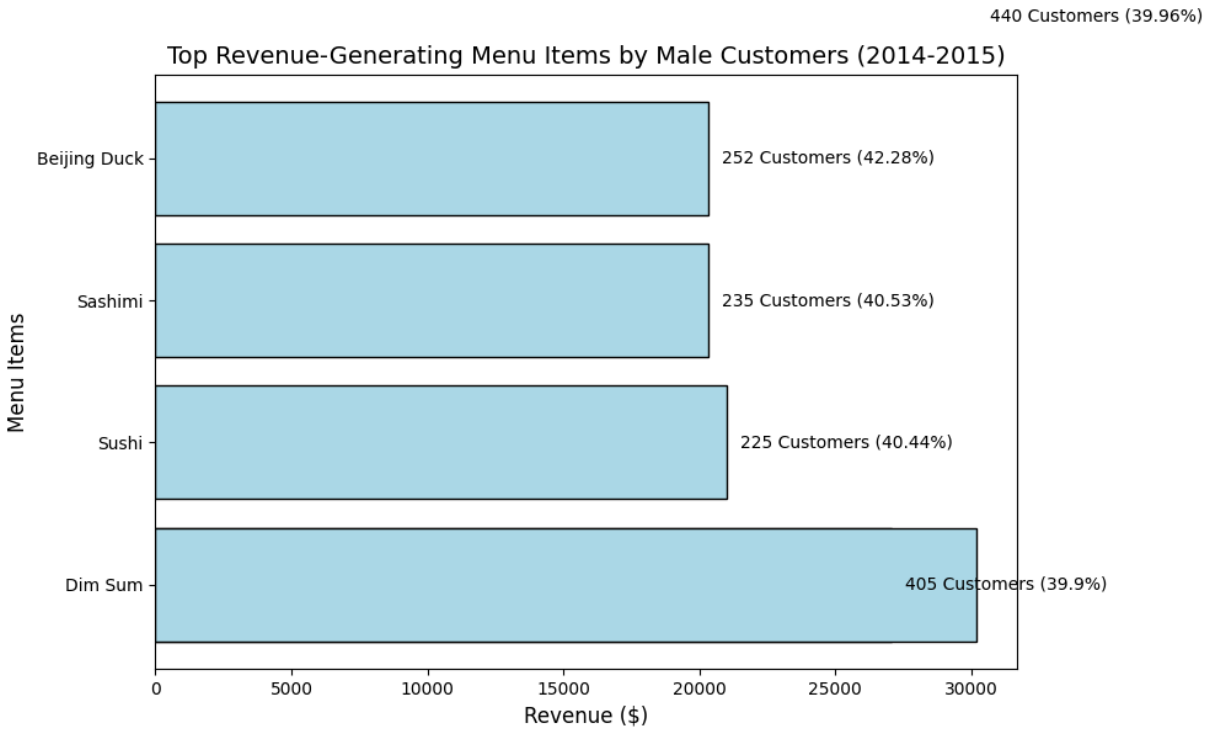
```
ACCEPT TargetQuarter CHAR FORMAT 'A02' PROMPT "Enter a quarter (1/2/3/4): " DEFAULT '1';
ACCEPT rankpick CHAR FORMAT 'A02' PROMPT "Enter a rank (1,2,3...): " DEFAULT '5';

-- Define column formats
COLUMN MenuItem FORMAT A40 HEADING 'Menu Item';
COLUMN Year FORMAT A4 HEADING 'Year';
COLUMN Quarter FORMAT A10 HEADING 'Quarter';
COLUMN Gender FORMAT A6 HEADING 'Gender';
COLUMN TotalRevenue FORMAT $999,999.99 HEADING 'Total Revenue';
COLUMN GenderCustomers FORMAT 999,999 HEADING 'Customers by Gender';
COLUMN GenderRevenuePercentage FORMAT A12 HEADING 'Revenue by Gender(%)';

-- Define title for the report
TTITLE CENTER 'Top &rankpick Revenue-Generating Menu Items by Gender, Year, and Quarter' -
       SKIP 1 CENTER 'For Quarter &TargetQuarter from Year &TargetYearS to Year &TargetYearE' -
       SKIP 1 RIGHT 'Date: ' _DATE SKIP 1 RIGHT 'Page ' FORMAT 999 SQL.PNO SKIP 2;

-- Add breaks on Year, Quarter, and Gender
BREAK ON Year SKIP 1 ON Quarter SKIP 1 ON Gender SKIP 1;

-- Create view for the main query
CREATE OR REPLACE VIEW RevenueBreakdown AS (
    SELECT
        M.FOODNDRINK AS MenuItem,
        D.cal_year AS Year,
        D.cal_quarter AS Quarter,
        C.CUSTOMER_GENDER AS Gender,
        COUNT(O.orderID) AS GenderCustomers,  -- Count of orders by gender
        SUM(O.unit_price * O.quantity) AS TotalRevenue,
        SUM(COUNT(O.orderID)) OVER (PARTITION BY M.FOODNDRINK, D.cal_year, D.cal_quarter) AS
TotalCustomers,  -- Total customers for each menu item
        SUM(SUM(O.unit_price * O.quantity)) OVER (PARTITION BY M.FOODNDRINK, D.cal_year,
D.cal_quarter) AS MenuItemRevenue,  -- Total revenue for each menu item
        ROW_NUMBER() OVER (PARTITION BY C.CUSTOMER_GENDER ORDER BY SUM(O.unit_price * O.quantity)
DESC) AS Rank  -- Rank menu items by revenue for each gender
    FROM
        Order_Facts O
        JOIN Menu_dim M ON O.menu_key = M.menu_key
        JOIN Customer_dim C ON O.customer_key = C.customer_key
        JOIN Date_dim D ON O.date_key = D.date_key
        JOIN Platform_dim PD ON O.Platform_key = PD.Platform_key
    WHERE
        D.cal_quarter = &TargetQuarter
        AND D.cal_year BETWEEN &TargetYearS AND &TargetYearE
        AND PD.PlatformName != 'In Stall'  -- Exclude 'In Stall' platform
    GROUP BY
        M.FOODNDRINK, D.cal_year, D.cal_quarter, C.CUSTOMER_GENDER
);
```

```
-- Final query to display top-ranked items
SELECT
    R.MenuItem,
    TO_CHAR(R.Year) AS Year,
    R.Gender,
    TO_CHAR(R.TotalRevenue, '$999,999.99') AS TotalRevenue,
    R.GenderCustomers,
    TO_CHAR((R.TotalRevenue / R.MenuItemRevenue) * 100, '90.99') || '%' AS Gender_Percentage
FROM
    RevenueBreakdown R
WHERE
    R.Rank <= &rankpick  -- Filter to show only the top-ranked items
ORDER BY
    Year,R.Gender;

-- Clear settings
CLEAR COLUMNS;
CLEAR BREAKS;
TTITLE OFF;
```

Output:

```
Enter a start year (YYYY):
Enter an end year (YYYY):
Enter a quarter (1/2/3/4):
Enter a rank (1,2,3...): 5
old  19:         D.cal_quarter = &TargetQuarter
new  19:         D.cal_quarter = 1
old  20:         AND D.cal_year BETWEEN &TargetYearS AND &TargetYearE
new  20:         AND D.cal_year BETWEEN 2014 AND 2015

View created.

old  11:     R.Rank <= &rankpick  -- Filter to show only the top-ranked items
new  11:     R.Rank <= 5  -- Filter to show only the top-ranked items

                                    Top 5 Revenue-Generating Menu Items by Gender, Year, and Quarter
                                                For Quarter 1 from Year 2014 to Year 2015
                                                                                                                Date: 15/09/2024
                                                                                                                      Page    1

Menu Item                           Year Gender Total Revenue           Customers by Gender GENDER_PERCENT
----------------------------------- ---- ------ ---------------------- ------------------- --------------
Dim Sum                             2014 F        $40,770.00                          613  60.10%
Beijing Duck                                      $31,724.00                          371  63.26%
Sushi                                             $30,930.00                          347  59.56%

Dim Sum                                  M        $27,067.50                          405  39.90%
Sushi                                             $21,000.00                          225  40.44%


Dim Sum                             2015 F        $45,360.00                          676  60.04%
Sashimi                                           $29,811.00                          336  59.47%

Sashimi                                  M        $20,320.50                          235  40.53%
Beijing Duck                                      $20,328.00                          252  42.28%
Dim Sum                                           $30,195.00                          440  39.96%
```





Base on the table and plot above we can know the top revenue menu item which split by Gender ,specific quarter Year and Quarter with in the rank of top revenue choose .it show the top revenue generating item in the company which can conduct gender-specific promotions to stimulate consumer spending,Most female customers like our popular foods more than male customers. Starting from here, we can hold many activities such as the half-price event for women on Mother's Day, or the event scales on Father's Day and Valentine's Day. We can make better plans for the event costs and future event scales based on the data from previous years.

3.3.2 Penalty and Contract Compliance Analysis with year Penalty Growth and risk assessment by specific Year

```
SET SERVEROUTPUT ON
CLEAR SCREEN
SET linesize 135
SET pagesize 190
ALTER SESSION SET NLS_DATE_FORMAT = 'dd/mm/yyyy';

-- Accept user input for start year, end year, and risk levels
ACCEPT StartYear CHAR FORMAT 'A4' PROMPT "Enter the start year (YYYY): " DEFAULT '2014';
ACCEPT EndYear CHAR FORMAT 'A4' PROMPT "Enter the end year (YYYY): " DEFAULT '2015';
ACCEPT HighRiskLevel NUMBER PROMPT "Enter the threshold for High Risk: " DEFAULT 7000;
ACCEPT MediumRiskLevel NUMBER PROMPT "Enter the threshold for Medium Risk: " DEFAULT 5000;

-- Define column formats
COLUMN TenantName FORMAT A25 HEADING 'Tenant Name';
COLUMN TotalPenalties FORMAT $999,999.99 HEADING 'Total Penalties';
COLUMN PenaltyCount FORMAT 999 HEADING 'Penalty Count';
COLUMN AvgPenalty FORMAT $999,999.99 HEADING 'Avg Penalty';
COLUMN Year FORMAT A5 HEADING 'Year';
COLUMN PenaltyGrowth FORMAT A15 HEADING 'Penalty Growth (%)';
COLUMN RiskLevel FORMAT A10 HEADING 'Risk Level';

-- Define title for the report
TTITLE CENTER 'Penalty and Compliance Analysis by Year' -
       SKIP 1 CENTER 'From Year &StartYear to Year &EndYear' -
       SKIP 1 RIGHT 'Date: ' _DATE SKIP 1 RIGHT 'Page ' FORMAT 999 SQL.PNO SKIP 2;

-- Add breaks on Year and TenantName
BREAK ON Year SKIP 1 ON TenantName SKIP 1;

-- Main query to get penalty data by year with additional analysis
WITH PenaltyData AS (
    SELECT
        B.tenantName AS TenantName,
        COUNT(C.penaltyAmount) AS PenaltyCount,
```

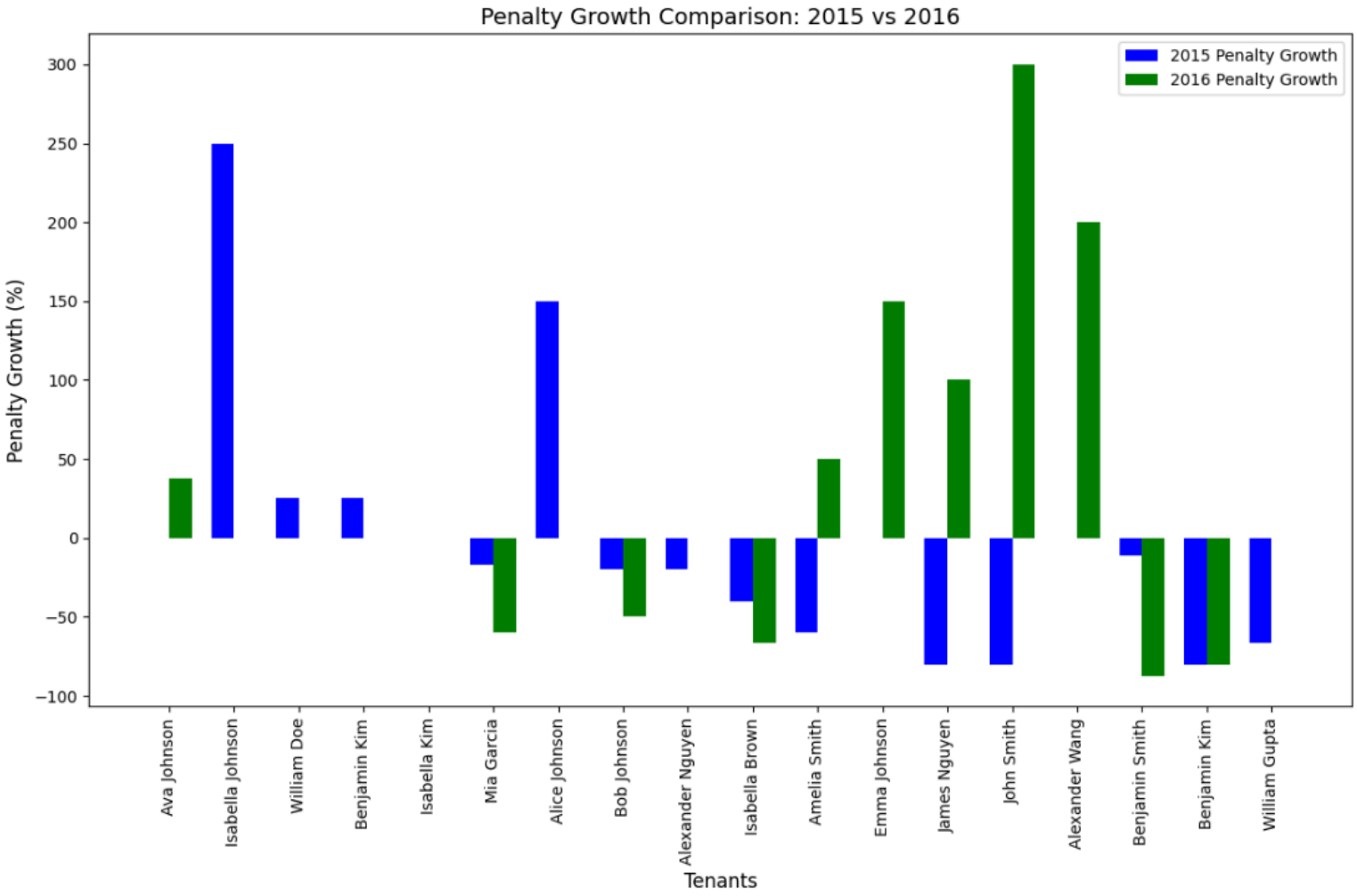
```
        SUM(C.penaltyAmount) AS TotalPenalties,
        AVG(C.penaltyAmount) AS AvgPenalty,
        EXTRACT(YEAR FROM D.cal_date) AS Year
    FROM
        contract_fact C
        JOIN Business_dim B ON C.business_key = B.business_key
        JOIN Date_dim D ON C.date_key = D.date_key
    WHERE
        D.cal_year BETWEEN &StartYear AND &EndYear
    GROUP BY
        B.tenantName, EXTRACT(YEAR FROM D.cal_date)
),
YearlyGrowth AS (
    -- Calculate year-over-year growth in penalties
    SELECT
        P1.TenantName,
        P1.Year,
        P1.TotalPenalties,
        P1.PenaltyCount,
        P1.AvgPenalty,
        CASE
            WHEN P2.TotalPenalties IS NULL THEN 'N/A'  -- No data for the previous year
            WHEN P1.TotalPenalties IS NULL THEN '000.00%'
            ELSE TO_CHAR(((P1.TotalPenalties - P2.TotalPenalties) / P2.TotalPenalties) * 100,
'999.99') || '%'
        END AS PenaltyGrowth
    FROM
        PenaltyData P1
    LEFT JOIN
        PenaltyData P2 ON P1.TenantName = P2.TenantName AND P1.Year = P2.Year + 1
    WHERE
        P1.Year BETWEEN &StartYear AND &EndYear
)
-- Final query to display the results with user-adjustable risk assessment
SELECT
    YG.TenantName,
    YG.TotalPenalties,
    YG.PenaltyCount,
    TO_CHAR(YG.AvgPenalty, '999,999.99') AS AvgPenalty,
    TO_CHAR(YG.Year) YEAR,
    YG.PenaltyGrowth,
    CASE
        WHEN YG.TotalPenalties > &HighRiskLevel THEN 'High'
        WHEN YG.TotalPenalties BETWEEN &MediumRiskLevel AND &HighRiskLevel THEN 'Medium'
        ELSE 'Low'
    END AS RiskLevel
FROM
    YearlyGrowth YG
ORDER BY
```

```
    YG.Year ASC, YG.TotalPenalties DESC;

-- Clear settings after execution
CLEAR COLUMNS;
CLEAR BREAKS;
TTITLE OFF;
```

Output:

```
Enter the start year (YYYY): 2014
Enter the end year (YYYY): 2016
Enter the threshold for High Risk: 900
Enter the threshold for Medium Risk: 400
old  13:         D.cal_year BETWEEN &StartYear AND &EndYear
new  13:         D.cal_year BETWEEN 2014 AND 2016
old  35:         P1.Year BETWEEN &StartYear AND &EndYear
new  35:         P1.Year BETWEEN 2014 AND 2016
old  46:         WHEN YG.TotalPenalties > &HighRiskLevel THEN 'High'
new  46:         WHEN YG.TotalPenalties >        900 THEN 'High'
old  47:         WHEN YG.TotalPenalties BETWEEN &MediumRiskLevel AND &HighRiskLevel THEN 'Medium'
new  47:         WHEN YG.TotalPenalties BETWEEN        400 AND        900 THEN 'Medium'

                                        Penalty and Compliance Analysis by Year
                                             From Year 2014 to Year 2016
                                                                                                                  Date: 15/09/2024
                                                                                                                         Page    1

Tenant Name               Total Penalties Penalty Count Avg Penalty            Year  Penalty Growth  Risk Level
------------------------- --------------- ------------- ---------------------- ----- --------------- ----------
Benjamin Smith                    $900.00             2      450.00            2014  N/A             Medium

Ava Johnson                       $700.00             2      350.00                  N/A             Medium

Mia Garcia                        $600.00             2      300.00                  N/A             Medium

John Smith                        $500.00             1      500.00                  N/A             Medium

Amelia Smith                      $500.00             1      500.00                  N/A             Medium

James Nguyen                      $500.00             1      500.00                  N/A             Medium

Isabella Brown                    $500.00             1      500.00                  N/A             Medium

Alexander Nguyen                  $500.00             1      500.00                  N/A             Medium

Isabella Kim                      $500.00             1      500.00                  N/A             Medium

Bob Johnson                       $500.00             1      500.00                  N/A             Medium

William Doe                       $400.00             1      400.00                  N/A             Medium

Benjamin Kim                      $400.00             1      400.00                  N/A             Medium

William Gupta                     $300.00             1      300.00                  N/A             Low

David Kim                         $200.00             1      200.00                  N/A             Low

Alice Johnson                     $200.00             1      200.00                  N/A             Low

Isabella Johnson                  $200.00             1      200.00                  N/A             Low

Emma Johnson                      $200.00             1      200.00                  N/A             Low
```

Benjamin Smith	$800.00	2	400.00	2015	-11.11%	Medium
Ava Johnson	$800.00	2	400.00		14.29%	Medium
Isabella Johnson	$700.00	2	350.00		250.00%	Medium
William Doe	$500.00	1	500.00		25.00%	Medium
Benjamin Kim	$500.00	1	500.00		25.00%	Medium
Isabella Kim	$500.00	1	500.00		.00%	Medium
Mia Garcia	$500.00	1	500.00		-16.67%	Medium
Alice Johnson	$500.00	1	500.00		150.00%	Medium
Bob Johnson	$400.00	1	400.00		-20.00%	Medium
Alexander Nguyen	$400.00	1	400.00		-20.00%	Medium
Isabella Brown	$300.00	1	300.00		-40.00%	Low
Amelia Smith	$200.00	1	200.00		-60.00%	Low
Emma Johnson	$200.00	1	200.00		.00%	Low
James Nguyen	$100.00	1	100.00		-80.00%	Low
William Gupta	$100.00	1	100.00		-66.67%	Low
John Smith	$100.00	1	100.00		-80.00%	Low
Alexander Wang	$100.00	1	100.00		N/A	Low
Ava Johnson	$1,100.00	3	366.67	2016	37.50%	High
Isabella Johnson	$700.00	2	350.00		.00%	Medium
Emma Johnson	$500.00	1	500.00		150.00%	Medium
William Doe	$500.00	1	500.00		.00%	Medium
Alice Johnson	$500.00	2	250.00		.00%	Medium
Isabella Kim	$500.00	1	500.00		.00%	Medium
John Smith	$400.00	1	400.00		300.00%	Medium
Alexander Wang	$300.00	1	300.00		200.00%	Low
Amelia Smith	$300.00	1	300.00		50.00%	Low
James Nguyen	$200.00	1	200.00		100.00%	Low



The table and bar chart above show the penalty growth percentages for tenants in 2014-2015-2016, highlighting the change in penalties compared to the previous period. This helps in identifying tenants with significant changes, both positive (increased penalties) and negative (reduced penalties).

Example like based on the graph , Tenants like Isabella Johnson and Alice Johnson show the highest penalty growth, we can focus on tenants with a history of non-compliance to implement corrective measures and improve their performance, Tenants like John Smith and Isabella kim have significant reductions in penalties, suggesting improved compliance.Tenants with a stable or no change in penalties, such as benjaminKim, may need monitoring for further trends.This can guide and investigate the underlying reasons for penalties to address systemic issues and prevent future non-compliance.

Furthermore ,we can Reward tenants with consistent compliance to encourage continued adherence to regulations. If there is a large demand for renting stalls, we can quickly and batch terminate the contractor and issue a warning.Not only that, we can also analyze it by year and quarter to optimize your penalty and compliance management efforts.

3.3.3 Customer Churn Risk Prediction

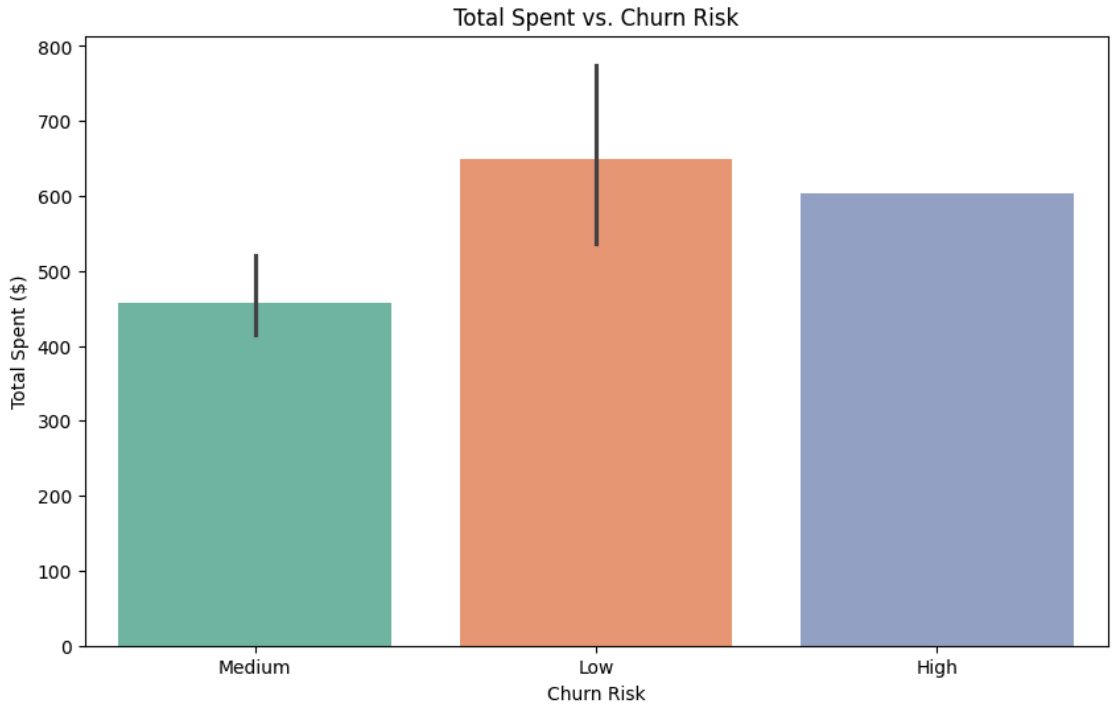
```
-- Enable user interaction and server output
SET SERVEROUTPUT ON
CLEAR SCREEN
SET linesize 100
SET pagesize 190
ALTER SESSION SET NLS_DATE_FORMAT = 'dd/mm/yyyy';

-- Accept user input for evaluation range, churn thresholds, and filters
ACCEPT StartDate CHAR FORMAT 'A10' PROMPT "Enter start date (DD/MM/YYYY): " DEFAULT '01/01/2014';
ACCEPT EndDate CHAR FORMAT 'A10' PROMPT "Enter end date (DD/MM/YYYY): " DEFAULT '31/12/2016';
ACCEPT TargetDate CHAR FORMAT 'A10' PROMPT "Enter target date (DD/MM/YYYY) to calculate churn
risk: " DEFAULT '1/1/2016';
ACCEPT HighRiskMonths NUMBER FORMAT '999' PROMPT "Enter months threshold for High risk (e.g., 6):
" DEFAULT '6';
ACCEPT MediumRiskMonths NUMBER FORMAT '999' PROMPT "Enter months threshold for Medium risk (e.g.,
3): " DEFAULT '3';
ACCEPT MinSpent NUMBER FORMAT '999999' PROMPT "Enter minimum total spent to include (e.g., 100):
" DEFAULT '200';
ACCEPT MinOrders NUMBER FORMAT '999' PROMPT "Enter minimum number of orders to include (e.g., 1):
" DEFAULT '7';

-- Define column formats for output
COLUMN CUSTOMER_ID FORMAT A10 HEADING 'Customer_ID';
COLUMN Last_Purchase_Date FORMAT A15 HEADING 'LastPurchaseDate';
COLUMN Total_Quantity FORMAT 999,999 HEADING 'Total Quantity';
COLUMN Total_Spent FORMAT $999,999.99 HEADING 'Total Spent';
COLUMN Total_Orders FORMAT 999 HEADING 'Total Orders';
COLUMN Churn_Risk_Level FORMAT A10 HEADING 'Churn Risk Level';

-- Define title for the report
TTITLE CENTER 'Customer Churn Risk Prediction' -
```

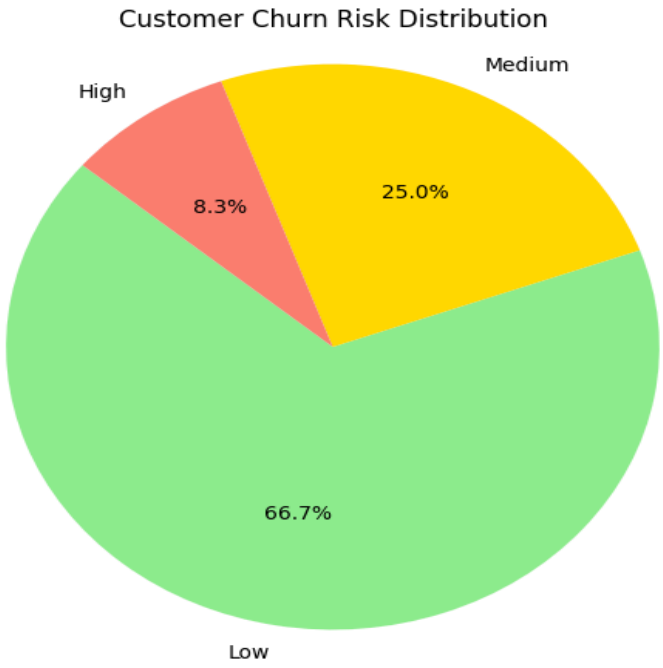
```
       SKIP 1 CENTER 'Based on Purchase Patterns and Recency' -
       SKIP 1 CENTER 'From &StartDate to &EndDate' -
       SKIP 1 CENTER 'TargetDate: &TargetDate ' -
       SKIP 1 RIGHT 'Date: ' _DATE SKIP 1 RIGHT 'Page ' FORMAT 999 SQL.PNO SKIP 2;

-- Main query for customer activity and churn risk prediction
WITH Customer_Activity AS (
    SELECT
        c.CUSTOMER_ID,
        MAX(d.cal_date) AS Last_Purchase_Date,
        SUM(o.quantity) AS Total_Quantity,
        SUM(o.quantity * o.unit_price) AS Total_Spent,
        COUNT(DISTINCT o.orderID) AS Total_Orders
    FROM
        Order_Facts o
    JOIN
        Customer_dim c ON o.CUSTOMER_KEY = c.CUSTOMER_KEY
    JOIN
        Date_dim d ON o.date_key = d.date_key
    WHERE
        d.cal_date BETWEEN TO_DATE('&StartDate', 'DD/MM/YYYY') AND TO_DATE('&EndDate',
'DD/MM/YYYY')
    GROUP BY
        c.CUSTOMER_ID
),
Churn_Risk AS (
    SELECT
        CUSTOMER_ID,
        Last_Purchase_Date,
        Total_Quantity,
        Total_Spent,
        Total_Orders,
        CASE
            -- Use the user-defined Target Date instead of SYSDATE to calculate churn risk
            WHEN Last_Purchase_Date < ADD_MONTHS(TO_DATE('&TargetDate', 'DD/MM/YYYY'),
-&HighRiskMonths) THEN 'High'
            WHEN Last_Purchase_Date BETWEEN ADD_MONTHS(TO_DATE('&TargetDate', 'DD/MM/YYYY'),
-&HighRiskMonths)
                AND ADD_MONTHS(TO_DATE('&TargetDate', 'DD/MM/YYYY'), -&MediumRiskMonths) THEN
'Medium'
            ELSE 'Low'
        END AS Churn_Risk_Level
    FROM
        Customer_Activity
    WHERE
        -- Apply filters for minimum total spent and minimum total orders
        Total_Spent >= &MinSpent
        AND Total_Orders >= &MinOrders
)
```

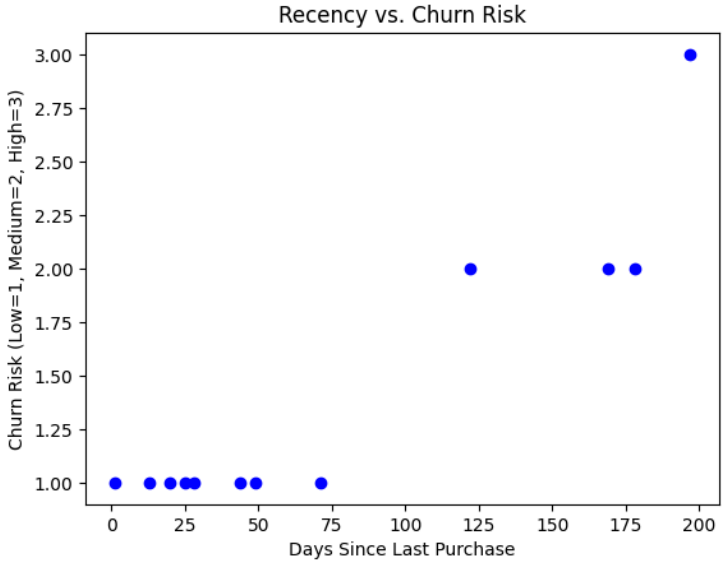
```
-- Final result selection and ordering
SELECT
    TO_CHAR(CUSTOMER_ID)as Customer_ID,
    TO_CHAR(Last_Purchase_Date, 'DD/MM/YYYY') AS Last_Purchase_Date,
    Total_Quantity,
    Total_Spent,
    Total_Orders,
    Churn_Risk_Level
FROM
    Churn_Risk
where Customer_ID >=1
ORDER BY
    Churn_Risk_Level DESC, Total_Spent DESC;

-- Clear settings after execution
CLEAR COLUMNS;
TTITLE OFF;
```

Output:

```
Enter start date (DD/MM/YYYY):
Enter end date (DD/MM/YYYY):
Enter target date (DD/MM/YYYY) to calculate churn risk:
Enter months threshold for High risk (e.g., 6):
Enter months threshold for Medium risk (e.g., 3):
Enter minimum total spent to include (e.g., 100):
Enter minimum number of orders to include (e.g., 1):
old  15:         d.cal_date BETWEEN TO_DATE('&StartDate', 'DD/MM/YYYY') AND TO_DATE('&EndDate', 'DD/MM/YYYY')
new  15:         d.cal_date BETWEEN TO_DATE('01/01/2014', 'DD/MM/YYYY') AND TO_DATE('31/12/2016', 'DD/MM/YYYY')
old  28:             WHEN Last_Purchase_Date < ADD_MONTHS(TO_DATE('&TargetDate', 'DD/MM/YYYY'), -&HighRiskMonths) THEN 'High'
new  28:             WHEN Last_Purchase_Date < ADD_MONTHS(TO_DATE('1/1/2016', 'DD/MM/YYYY'), -         6) THEN 'High'
old  29:             WHEN Last_Purchase_Date BETWEEN ADD_MONTHS(TO_DATE('&TargetDate', 'DD/MM/YYYY'), -&HighRiskMonths)
new  29:             WHEN Last_Purchase_Date BETWEEN ADD_MONTHS(TO_DATE('1/1/2016', 'DD/MM/YYYY'), -         6)
old  30:                 AND ADD_MONTHS(TO_DATE('&TargetDate', 'DD/MM/YYYY'), -&MediumRiskMonths) THEN 'Medium'
new  30:                 AND ADD_MONTHS(TO_DATE('1/1/2016', 'DD/MM/YYYY'), -         3) THEN 'Medium'
old  37:         Total_Spent >= &MinSpent
new  37:         Total_Spent >=        200
old  38:         AND Total_Orders >= &MinOrders
new  38:         AND Total_Orders >=          7

                                Customer Churn Risk Prediction
                              Based on Purchase Patterns and Recency
                                   From 01/01/2014 to 31/12/2016
                                       TargetDate: 1/1/2016
                                                                                Date: 15/09/2024
                                                                                       Page    1

Customer_I LastPurchaseDat Total Quantity  Total Spent Total Orders Churn Risk
---------- -------------- -------------- ------------ ------------ ----------
227787     07/07/2015                 41      $520.25            7 Medium
275383     16/07/2015                 31      $434.40            7 Medium
202970     01/09/2015                 26      $415.70            7 Medium
149876     12/12/2015                 68      $904.45            7 Low
259577     31/12/2015                 64      $888.20            7 Low
203551     19/12/2015                 49      $792.50            7 Low
282669     22/10/2015                 43      $581.25            7 Low
314688     18/11/2015                 48      $531.55            7 Low
223460     04/12/2015                 42      $530.25            8 Low
322419     07/12/2015                 35      $505.50            7 Low
286555     13/11/2015                 29      $458.15            7 Low
168584     18/06/2015                 47      $603.55            7 High

12 rows selected.
```







From the table and graph above, we observe that it is designed to help businesses identify customers at risk of churning based on their purchasing behavior and engagement patterns over a specified period. It generates a report with customer churn risk levels and filters based on minimum total spent and total orders, which enables businesses to make data-driven decisions regarding retention strategies.

The goal of this query is to segment customers into different risk categories (High, Medium, and Low) based on how long it has been since their last purchase and their overall activity during a specified date range. It enable user Input for Customization:
Like Start and End Dates Allows decision-makers to analyze customer behavior during a specific period.
Target Date: This input provides flexibility for calculating churn risk based on a date of user choosing, enabling forward-looking or retrospective analysis.
Risk Thresholds in month Allows the business to define its own thresholds for categorizing High and Medium churn risk.
Minimum Spend and Minimum Orders: These filters ensure that the analysis only focuses on valuable customers who meet a minimum threshold of purchases and spending, avoiding over-investment in low-value customers.The decisions that we can make are when we detect High-risk customers may need immediate attention through personalized retention offers (e.g., discounts, loyalty rewards), while medium-risk customers might benefit from targeted re-engagement campaigns. Low-risk customers may not require immediate action but should be nurtured for long-term loyalty.
This will make sure that the retention efforts are focused on customers with significant lifetime value.
Business Implication: Prioritizing resources on high-value customers improves the return on investment on retention campaigns.
And for the pie chart we can easy to know the the ratio of churn risk of the target customer group
This query not only helps in predicting churn risk but also provides actionable insights for decision-makers. By customizing the churn risk parameters and filtering criteria, businesses can better focus their retention strategies on high-value customers, thereby maximizing the efficiency of retention efforts.