FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

Bachelor of Computer Science (Honours) in Data Science

BAIT 3003 Data Warehouse (202305)

Student Name	Student ID	Signature
TANG SHARREN	21WMR01086	sharren
THAM HIU HUEN	21WMR13343	tham
KOONG JIE LUM	21WMR13414	koong
TAN MEI YIN	21WMR07586	TAN.

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. ● Must include ONE Type 2 SCD ● 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. ● Variety of techniques necessary to achieve the correct data loading	● 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

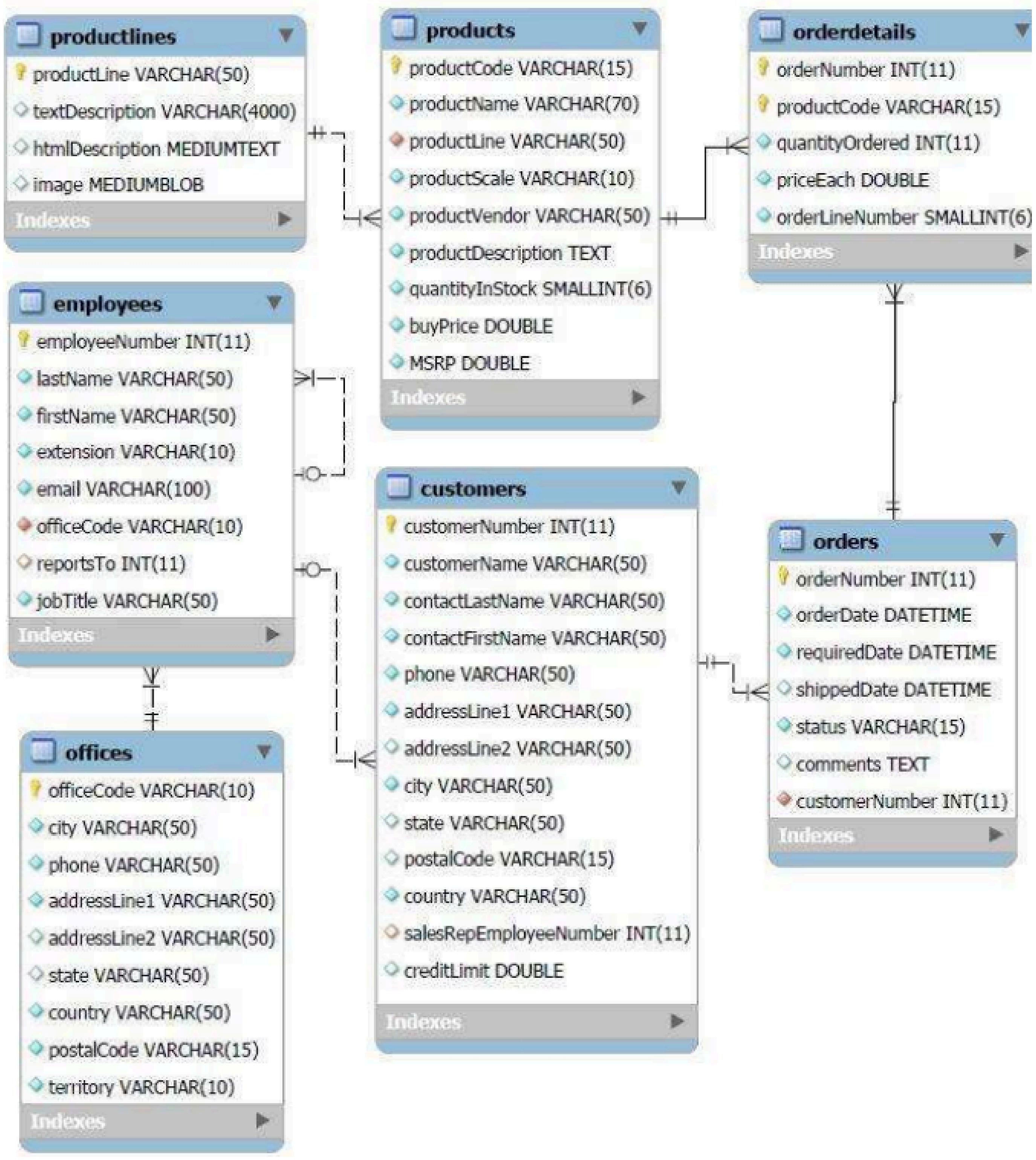
<u>Group Member:</u>		Task 3 marks	Total marks
1.	<u>Tang Sharren</u>	(…………….)	( )
2.	<u>Tham Hiu Huen</u>	(.....................)	( )
3.	<u>Koong Jie Lum</u>	(……………...)	( )
4.	<u>Tan Mei Yin</u>	(……………..)	( )

Table of Content

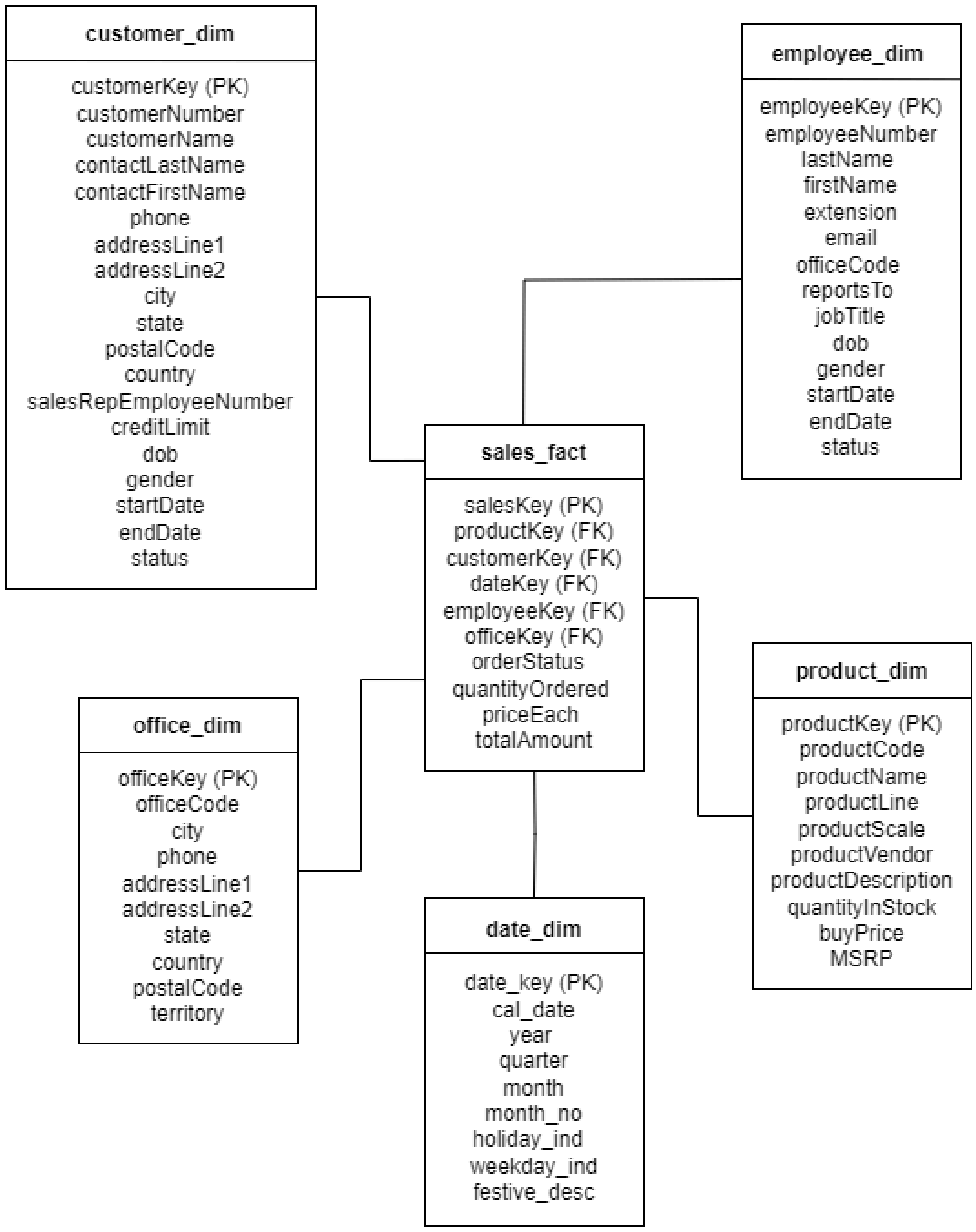
Chapter 1 Design of Data Warehouse

1.1 Logical Design

<u>1.1.1 Original Database</u>



1.1.2 Star Schema Dimension and Fact Tables



1.2 Physical Design

1.2.1 Original Database

```
CREATE TABLE offices (
  officeCode VARCHAR(10) NOT NULL,
  city VARCHAR(50) NOT NULL,
  phone VARCHAR(50) NOT NULL,
  addressLine1 VARCHAR(50) NOT NULL,
  addressLine2 VARCHAR(50) DEFAULT NULL,
  state VARCHAR(50) DEFAULT NULL,
  country VARCHAR(50) NOT NULL,
  postalCode VARCHAR(15) NOT NULL,
  territory VARCHAR(10) NOT NULL,
  PRIMARY KEY (officeCode)
);

CREATE TABLE productlines (
  productLine VARCHAR(50) NOT NULL,
  textDescription VARCHAR(4000) DEFAULT NULL,
  htmlDescription VARCHAR(1000),
  image BLOB,
  PRIMARY KEY (productLine)
);

CREATE TABLE products (
  productCode VARCHAR(15) NOT NULL,
  productName VARCHAR(70) NOT NULL,
  productLine VARCHAR(50) NOT NULL,
  productScale VARCHAR(10) NOT NULL,
  productVendor VARCHAR(50) NOT NULL,
  productDescription VARCHAR(4000) NOT NULL,
  quantityInStock NUMBER(4) NOT NULL,
  buyPrice NUMBER(7,2) NOT NULL,
  MSRP NUMBER(7,2) NOT NULL,
  PRIMARY KEY (productCode),
  FOREIGN KEY (productLine) REFERENCES productlines(productLine)
);

CREATE TABLE employees (
    employeeNumber NUMBER(11) NOT NULL,
    lastName VARCHAR2(50) NOT NULL,
    firstName VARCHAR2(50) NOT NULL,
    extension VARCHAR2(10),
    email VARCHAR2(100),
    officeCode VARCHAR2(10),
    reportsTo NUMBER(11) DEFAULT NULL,
    jobTitle VARCHAR2(50),
    PRIMARY KEY (employeeNumber),
    FOREIGN KEY (officeCode) REFERENCES offices(officeCode)
);

CREATE TABLE customers (
  customerNumber NUMBER(11) NOT NULL,
  customerName VARCHAR(50) NOT NULL,
  contactLastName VARCHAR(50) NOT NULL,
  contactFirstName VARCHAR(50) NOT NULL,
  phone VARCHAR(50) NOT NULL,
  addressLine1 VARCHAR(50) NOT NULL,
  addressLine2 VARCHAR(50) DEFAULT NULL,
  city VARCHAR(50) NOT NULL,
  state VARCHAR(50) DEFAULT NULL,
  postalCode VARCHAR(15) DEFAULT NULL,
  country VARCHAR(50) NOT NULL,
  salesRepEmployeeNumber NUMBER(11) DEFAULT NULL,
  creditLimit NUMBER(9,2) DEFAULT NULL,
  PRIMARY KEY (customerNumber),
  FOREIGN KEY (salesRepEmployeeNumber) REFERENCES
employees(employeeNumber)
);

CREATE TABLE orders (
  orderNumber NUMBER(11) NOT NULL,
  orderDate DATE NOT NULL,
  requiredDate DATE NOT NULL,
  shippedDate DATE DEFAULT NULL,
  status VARCHAR(15) NOT NULL,
  comments VARCHAR(500),
  customerNumber NUMBER(11) NOT NULL,
  PRIMARY KEY (orderNumber),
  FOREIGN KEY (customerNumber) REFERENCES customers(customerNumber)
);

CREATE TABLE orderdetails (
  orderNumber NUMBER(11) NOT NULL,
  productCode VARCHAR(15) NOT NULL,
  quantityOrdered NUMBER(4) NOT NULL,
  priceEach NUMBER(7,2) NOT NULL,
  orderLineNumber NUMBER(3) NOT NULL,
  PRIMARY KEY (orderNumber, productCode),
  FOREIGN KEY (orderNumber) REFERENCES orders(orderNumber),
  FOREIGN KEY (productCode) REFERENCES products(productCode)
);
```

1.2.2 Star Schema Dimension and Fact Tables

```
CREATE TABLE date_dim (
       date_key             NUMBER NOT NULL,
       cal_date             DATE NOT NULL,
       year                 NUMBER(4) NOT NULL,
       quarter              CHAR(2) NOT NULL,
       month                CHAR(15) NOT NULL,
       holiday_ind          CHAR(1) NOT NULL,
       weekday_ind          CHAR(1) NOT NULL,
       festive_desc         VARCHAR(10),
       PRIMARY KEY (date_key)
);
```

```
CREATE TABLE customer_dim (
    customerKey         NUMBER(11) NOT NULL,
    customerNumber      NUMBER(11) NOT NULL,
    customerName        VARCHAR(50) NOT NULL,
    contactLastName     VARCHAR(50) NOT NULL,
    contactFirstName    VARCHAR(50) NOT NULL,
    phone               VARCHAR(50) NOT NULL,
    addressLine1        VARCHAR(50) NOT NULL,
    addressLine2        VARCHAR(50),
    city                VARCHAR(50) NOT NULL,
    state               VARCHAR(50),
    postalCode          VARCHAR(15),
    country             VARCHAR(50) NOT NULL,
    salesRepEmployeeNumber NUMBER(11),
    creditLimit         NUMBER(9,2),
    dob                 DATE        NOT NULL,
    gender              CHAR(1)     NOT NULL,
    startDate           DATE,
    endDate             DATE,
    status              VARCHAR(15) NOT NULL,--DEL/ACT
    PRIMARY KEY (customerKey)
);
```

```
CREATE TABLE employee_dim (
    employeeKey         NUMBER(11) NOT NULL,
    employeeNumber      NUMBER(11) NOT NULL,
    lastName            VARCHAR(50) NOT NULL,
    firstName           VARCHAR(50) NOT NULL,
    extension           VARCHAR(10),
    email               VARCHAR(100),
    officeCode          VARCHAR(10),
    reportsTo           NUMBER(11),
    jobTitle            VARCHAR(50),
    dob                 DATE        NOT NULL,
    gender              CHAR(1)     NOT NULL,
    startDate           DATE,
    endDate             DATE,
    status              VARCHAR(15) NOT NULL,--DEL/ACT
    PRIMARY KEY (employeeKey)
);
```

```
CREATE TABLE product_dim (
    productKey          NUMBER(11) NOT NULL,
    productCode         VARCHAR(15) NOT NULL,
    productName         VARCHAR(70) NOT NULL,
    productLine         VARCHAR(50) NOT NULL,
    productScale        VARCHAR(10) NOT NULL,
    productVendor       VARCHAR(50) NOT NULL,
    productDescription  VARCHAR(4000) NOT NULL,
    quantityInStock     NUMBER(4) NOT NULL,
    buyPrice            NUMBER(7,2) NOT NULL,
    MSRP                NUMBER(7,2) NOT NULL,
    PRIMARY KEY (productKey)
);
```

```
CREATE TABLE office_dim (
    officeKey       NUMBER(11) NOT NULL,
    officeCode         VARCHAR(10) NOT NULL,
    city               VARCHAR(50) NOT NULL,
    phone              VARCHAR(50) NOT NULL,
    addressLine1   VARCHAR(50) NOT NULL,
    addressLine2   VARCHAR(50) DEFAULT NULL,
    state              VARCHAR(50) DEFAULT NULL,
    country            VARCHAR(50) NOT NULL,
    postalCode         VARCHAR(15) NOT NULL,
    territory    VARCHAR(10) NOT NULL,
    PRIMARY KEY (officeKey)
);
```

```
CREATE TABLE sales_fact (
    salesKey            NUMBER(11) NOT NULL,
    productKey          NUMBER(11) NOT NULL,
    customerKey         NUMBER(11) NOT NULL,
    dateKey             NUMBER NOT NULL,
    employeeKey         NUMBER(11) NOT NULL,
    officeKey           NUMBER(11) NOT NULL,
    orderStatus         VARCHAR(15) NOT NULL,
    quantityOrdered     NUMBER(4) NOT NULL,
    priceEach           NUMBER(7,2) NOT NULL,
    totalAmount         NUMBER(10,2) NOT NULL,
    PRIMARY KEY (salesKey),
    FOREIGN KEY (productKey) REFERENCES product_dim(productKey),
    FOREIGN KEY (customerKey) REFERENCES customer_dim(customerKey),
    FOREIGN KEY (dateKey) REFERENCES date_dim(date_key),
    FOREIGN KEY (employeeKey) REFERENCES employee_dim(employeeKey),
    FOREIGN KEY (officeKey) REFERENCES office_dim(officeKey)
);
```

Chapter 2 Extract, Transform, Load Process

2.1 Script for initial loading

2.1.1 For date_dim

```
(i)Change all dates in orders table to be starting from Jan 2021
till May 2023 instead
```

```
-- Update order dates to bring them forward to the current date
-- 2023-2005 = 18 years
-- 18 years * 12 months = 216 months
UPDATE orders
SET orderDate = ADD_MONTHS(orderDate, 216),
    requiredDate = ADD_MONTHS(requiredDate, 216),
    shippedDate = ADD_MONTHS(shippedDate, 216);
```

```
-- Drop the sequence if it already exists
DROP SEQUENCE date_seq;

-- Create a new sequence for date_dim
CREATE SEQUENCE date_seq
START WITH 10001
INCREMENT BY 1;

declare
   v_startDate date;
   v_endDate date;
   v_cal_date date;
   v_year number(4);
   v_quarter char(2);
   v_month varchar(15);
   v_holiday_ind char(1);
   v_weekday_ind  char(1);
   v_festive varchar(10);

begin
   v_holiday_ind := 'N';

   -- Set v_startDate and v_endDate based on orders table
   select min(orderDate), max(orderDate) into v_startDate,
v_endDate from orders;

   WHILE (v_startDate <= v_endDate) LOOP
      v_cal_date := v_startDate;
      v_year := TO_NUMBER(TO_CHAR(v_startDate, 'YYYY'));
      v_quarter := CASE
                     WHEN TO_NUMBER(TO_CHAR(v_startDate, 'MM'))
BETWEEN 1 AND 3 THEN 'Q1'
```

```
                     WHEN TO_NUMBER(TO_CHAR(v_startDate, 'MM'))
BETWEEN 4 AND 6 THEN 'Q2'
                     WHEN TO_NUMBER(TO_CHAR(v_startDate, 'MM'))
BETWEEN 7 AND 9 THEN 'Q3'
                     WHEN TO_NUMBER(TO_CHAR(v_startDate, 'MM'))
BETWEEN 10 AND 12 THEN 'Q4'
                  END;
      v_month := TO_CHAR(v_startDate, 'Month');
      v_weekday_ind := CASE
                          WHEN TO_CHAR(v_startDate, 'D') BETWEEN 2
AND 6 THEN 'Y'-- Weekday (Monday to Friday)
                          ELSE 'N' -- Weekend (Saturday and Sunday)
                        END;


       insert into date_dim values(date_seq.nextval,
                                   v_cal_date,
                                   v_year,
                                   v_quarter,
                                   v_month,
                                   v_holiday_ind ,
                                   v_weekday_ind ,
                                   null);

      v_startDate := v_startDate + 1;
   end loop;

end;
/
```

2.1.2 For customer_dim

a)Assign random DOB and state & city

```
 (i)Generate random DOB and store into a table
```

```
--Generate DOB for customer
drop table Gen_DOB;

create table Gen_DOB
(row_id number,
dob_date date);

set serveroutput on
create or replace procedure proc_gen_DOB(in_start_Date IN varchar,
                   in_end_Date IN varchar) IS
  v_startDate date;
  v_endDate   date;
  v_counter number :=1; --so that row_id is starting from 1

begin
   v_startDate:=to_date(in_start_Date,'dd/mm/yyyy');
   v_endDate  :=to_date(in_end_Date,'dd/mm/yyyy');

   while (v_startDate<=v_endDate) loop
       insert into Gen_DOB values(v_counter,v_startDate);
       v_counter:=v_counter+1;
       v_startDate:= v_startDate+1;

   end loop;
dbms_output.put_line('Count is ' ||v_counter);
end;
/
```

```
exec proc_gen_DOB('01/01/1950','01/01/2005');
--Count is 20090
```

(ii)Create state & city table

```
DROP TABLE StateAndCity;
CREATE TABLE StateAndCity(
  StateAndCityID        NUMBER(5),
  City                  VARCHAR(30)   NOT NULL,
  State                 VARCHAR(30)   NOT NULL,
PRIMARY KEY (StateAndCityID)
);


INSERT INTO StateAndCity VALUES (10001, 'Johor Bahru', 'Johor');
INSERT INTO StateAndCity VALUES (10002, 'Kluang', 'Johor');
INSERT INTO StateAndCity VALUES (10003, 'Kota Tinggi', 'Johor');
INSERT INTO StateAndCity VALUES (10004, 'Alor Setar', 'Kedah');
INSERT INTO StateAndCity VALUES (10005, 'Langkawi', 'Kedah');
INSERT INTO StateAndCity VALUES (10006, 'Bunut Payong',
'Kelantan');
INSERT INTO StateAndCity VALUES (10007, 'Melaka', 'Melaka');
INSERT INTO StateAndCity VALUES (10008, 'Port Dickson', 'Negeri
Sembilan');
INSERT INTO StateAndCity VALUES (10009, 'Seremban', 'Negeri
Sembilan');
INSERT INTO StateAndCity VALUES (10010, 'Genting Highlands',
'Pahang');
INSERT INTO StateAndCity VALUES (10011, 'Kuala Lipis', 'Pahang');
INSERT INTO StateAndCity VALUES (10012, 'Kuantan', 'Pahang');
INSERT INTO StateAndCity VALUES (10013, 'Mentakab', 'Pahang');
INSERT INTO StateAndCity VALUES (10014, 'Bidor', 'Perak');
INSERT INTO StateAndCity VALUES (10015, 'Ipoh', 'Perak');
INSERT INTO StateAndCity VALUES (10016, 'Kuala Perlis', 'Perlis');
INSERT INTO StateAndCity VALUES (10017, 'Bukit Mertajam', 'Pulau
Pinang');
INSERT INTO StateAndCity VALUES (10018, 'Butterworth', 'Pulau
Pinang');
INSERT INTO StateAndCity VALUES (10019, 'GeorgeTown', 'Pulau
Pinang');
INSERT INTO StateAndCity VALUES (10020, 'Perai', 'Pulau Pinang');
INSERT INTO StateAndCity VALUES (10021, 'Pulau Tikus', 'Pulau
Pinang');
INSERT INTO StateAndCity VALUES (10022, 'Seberang Perai', 'Pulau
Pinang');
```

```
INSERT INTO StateAndCity VALUES (10023, 'Kota Kinabalu', 'Sabah');
INSERT INTO StateAndCity VALUES (10024, 'Tawau', 'Sabah');
INSERT INTO StateAndCity VALUES (10025, 'Kapit', 'Sarawak');
INSERT INTO StateAndCity VALUES (10026, 'Kuching', 'Sarawak');
INSERT INTO StateAndCity VALUES (10027, 'Miri', 'Sarawak');
INSERT INTO StateAndCity VALUES (10028, 'Sibu', 'Sarawak');
INSERT INTO StateAndCity VALUES (10029, 'Ampang', 'Selangor');
INSERT INTO StateAndCity VALUES (10030, 'Balakong', 'Selangor');
INSERT INTO StateAndCity VALUES (10031, 'Banting', 'Selangor');
INSERT INTO StateAndCity VALUES (10032, 'Kajang', 'Selangor');
INSERT INTO StateAndCity VALUES (10033, 'Klang', 'Selangor');
INSERT INTO StateAndCity VALUES (10034, 'Petaling Jaya',
'Selangor');
INSERT INTO StateAndCity VALUES (10035, 'Rawang', 'Selangor');
INSERT INTO StateAndCity VALUES (10036, 'Sepang', 'Selangor');
INSERT INTO StateAndCity VALUES (10037, 'Seri Kembangan',
'Selangor');
INSERT INTO StateAndCity VALUES (10038, 'Shah Alam', 'Selangor');
INSERT INTO StateAndCity VALUES (10039, 'Subang Jaya', 'Selangor');
INSERT INTO StateAndCity VALUES (10040, 'Cheras', 'Wilayah
Persekutuan');
INSERT INTO StateAndCity VALUES (10041, 'Kuala Lumpur', 'Wilayah
Persekutuan');

select count(*) from StateAndCity;
```

(iii)To generate > 10,000 records in the temp_cust_unique table with unique customerNumber values starting from 500 and incrementing by 1,

```
DROP TABLE temp_cust;
-- Create a new table to hold the unique customer data
CREATE TABLE temp_cust AS SELECT * FROM customers WHERE 1=0;

-- Create a sequence to generate unique customer numbers starting
from 500
drop sequence cust_number_sequence;
CREATE SEQUENCE cust_number_sequence
  START WITH 500
  INCREMENT BY 1;

-- Loop to insert 10,000 records into the temp_cust table with
unique customer numbers
DECLARE
-- Initialize v_record_count
  v_counter NUMBER := 0;
BEGIN
  WHILE v_counter < 100 LOOP
    INSERT INTO temp_cust (customerNumber, customerName,
contactLastName, contactFirstName, phone, addressLine1,
addressLine2, city, state, postalCode, country,
salesRepEmployeeNumber, creditLimit)
    SELECT cust_number_sequence.NEXTVAL, customerName,
contactLastName, contactFirstName, phone, addressLine1,
addressLine2, city, state, postalCode, country,
salesRepEmployeeNumber, creditLimit
    FROM customers;
    v_counter := v_counter + 1;
  END LOOP;
END;
/

-- Check the result
SELECT COUNT(*) as "Records in temp_cust" FROM temp_cust;
```

(iv) Assign the random DOB, gender, state & city to new_cust
Generate updated records for some customer and assign random endDate(must be > startDate), assign the old record’s ‘status’ as del

```
DROP TABLE new_cust;
CREATE TABLE new_cust AS SELECT * FROM customer_dim WHERE 1=0;

drop sequence cust_sequence;
create sequence cust_sequence
start with 100001
increment by 1;
DECLARE
  CURSOR cust_cur IS
    SELECT *
    FROM temp_cust
    ORDER BY customerNumber;

  t_rec cust_cur%ROWTYPE;

  v_random_rowID NUMBER;
  v_dob DATE;
  v_ID NUMBER;
  v_city VARCHAR(30);
  v_state VARCHAR(30);
  v_gender CHAR(1);
  v_startDate DATE;
  v_endDate DATE;
  v_status VARCHAR2(3); -- Assuming status is a VARCHAR2(3) field

  -- Variables for controlling the percentage of records with
endDate
  v_records_count NUMBER := 0;
  v_records_with_endDate NUMBER;
  v_percentage NUMBER := 20; -- Change this to set the desired
percentage

BEGIN
  -- Get the total number of records
  SELECT COUNT(*) as "Records in temp_cust" INTO v_records_count
FROM temp_cust;

  -- Calculate the number of records with endDate
  v_records_with_endDate := ROUND(v_records_count * v_percentage /
100);
```

```
    FOR t_rec IN cust_cur LOOP

      -- Assign DOB randomly
      v_random_rowID := TRUNC(DBMS_RANDOM.VALUE(1, 20091));
      SELECT dob_date INTO v_dob
      FROM Gen_DOB
      WHERE row_id = v_random_rowID;

      -- Assign state and city randomly
      -- Generate a random v_ID within a valid range
      v_ID := TRUNC(DBMS_RANDOM.VALUE(10001, 10042));

      -- Query the StateAndCity table to get city and state for the
generated v_ID
      SELECT city, state INTO v_city, v_state
      FROM StateAndCity
      WHERE StateAndCityID = v_ID;

      -- Assign gender randomly
      IF (MOD(v_random_rowID, 2) = 0) THEN
        v_gender := 'M';
      ELSE
        v_gender := 'F';
      END IF;

      -- Generate a random number of days to add to the start date
      v_startDate := TO_DATE('01/01/2020', 'DD/MM/YYYY') +
TRUNC(DBMS_RANDOM.VALUE(1, TO_DATE('01/01/2022', 'DD/MM/YYYY') -
TO_DATE('01/01/2000', 'DD/MM/YYYY')));

      -- Determine if this record should have an endDate
      IF v_records_with_endDate > 0 THEN
        -- Generate a random endDate greater than startDate
        v_endDate := v_startDate + TRUNC(DBMS_RANDOM.VALUE(1, 365));
-- Adjust the range as needed
        v_records_with_endDate := v_records_with_endDate - 1;

        -- Assign status as 'DEL' for records with endDate
        v_status := 'DEL';
      ELSE
        v_endDate := NULL; -- No endDate for this record
        v_status := 'ACT';  -- Status is active for records without
endDate
      END IF;

      -- Insert into new_cust
      INSERT INTO new_cust (
        customerKey,
        customerNumber,
        customerName,
        contactLastName,
        contactFirstName,
        phone,
        addressLine1,
        addressLine2,
        city,
        state,
        postalCode,
        country,
        salesRepEmployeeNumber,
        creditLimit,
        dob,
        gender,
        startDate,
        endDate,
        status
      )
      VALUES (
        cust_sequence.NEXTVAL,
        t_rec.customerNumber,
        t_rec.customerName,
        t_rec.contactLastName,
        t_rec.contactFirstName,
        t_rec.phone,
        t_rec.addressLine1,
        t_rec.addressLine2,
        v_city,
        v_state,
        t_rec.postalCode,
        t_rec.country,
        t_rec.salesRepEmployeeNumber,
        t_rec.creditLimit,
        v_dob,
        v_gender,
        v_startDate,
        v_endDate,
        v_status
      );
    END LOOP;
  END;
  /
```

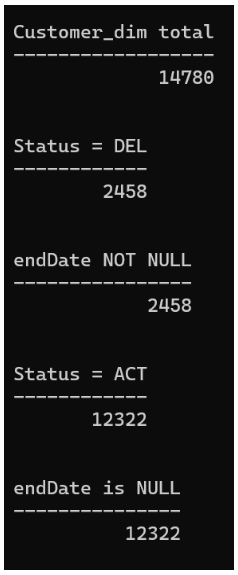
```
SELECT COUNT(*) as "Status = DEL(new_cust)"
FROM new_cust
WHERE status = 'DEL';
-- 20% * 12,200 = 2,440
```

(v) Generate random phone number with length of 10 for customer records with status 'DEL'

```
-- Create the updated_cust table if it doesn't already exist
DROP TABLE update_cust;
CREATE TABLE update_cust AS
SELECT *
FROM new_cust
WHERE status = 'DEL';

-- Update the status, endDate and phone for all rows in the
updated_cust table
UPDATE update_cust
SET status = 'ACT',
    endDate = NULL,
    phone = LPAD(TRUNC(DBMS_RANDOM.VALUE(0, 9999999999)), 10, '0');

-- Insert all rows from update_cust into new_cust with
cust_sequence.NEXTVAL as customerKey
```

```
INSERT INTO new_cust (
  customerKey,
  customerNumber,
  customerName,
  contactLastName,
  contactFirstName,
  phone,
  addressLine1,
  addressLine2,
  city,
  state,
  postalCode,
  country,
  salesRepEmployeeNumber,
  creditLimit,
  dob,
  gender,
  startDate,
  endDate,
  status
)
SELECT
  cust_sequence.NEXTVAL,
  customerNumber,
  customerName,
  contactLastName,
  contactFirstName,
  phone,
  addressLine1,
  addressLine2,
  city,
  state,
  postalCode,
  country,
  salesRepEmployeeNumber,
  creditLimit,
  dob,
  gender,
  startDate,
  endDate,
  status
FROM update_cust;
```

```
Output to prove applying Type 2 SCD in customer_dim
--Old record: With an endDate and status is 'DEL'
--Active record: Without endDate and status is 'ACT'
```



```
Customer_dim total
------------------
             14780


Status = DEL
------------
        2458


endDate NOT NULL
----------------
            2458


Status = ACT
------------
       12322


endDate is NULL
---------------
          12322
```

2.1.3 For employee_dim

```
SET LINESIZE 300;
SET PAGESIZE 300;

DROP TABLE employee_dim;

CREATE TABLE employee_dim (
    employeeKey         NUMBER(11) NOT NULL,
    employeeNumber      NUMBER(11) NOT NULL,
    lastName            VARCHAR(50) NOT NULL,
    firstName           VARCHAR(50) NOT NULL,
    extension           VARCHAR(10),
    email               VARCHAR(100),
    officeCode          VARCHAR(10),
    reportsTo           NUMBER(11),
    jobTitle            VARCHAR(50),
    dob                 DATE        NOT NULL,
    gender              CHAR(1)     NOT NULL,
    startDate           DATE,
    endDate             DATE,
    status              VARCHAR(15) NOT NULL,--DEL/ACT
    PRIMARY KEY (employeeKey)
);
```

```
-- (i)Generate random DOB and store into a table
```

```
-- In employee_dim steps, we have done exec
proc_gen_DOB('01/01/1950','01/01/2005');
-- Minimum row_id
SELECT MIN(row_id) AS "Min. row_id in GEN_DOB"
FROM Gen_DOB;

-- Maximum row_id
SELECT MAX(row_id) AS "Max. row_id in GEN_DOB"
FROM Gen_DOB;
--Min is 1, max is 20090
```

```
--(ii)Save unique employee number into a table GEN_EMP
```

```
-- Create the GEN_EMP table
DROP TABLE GEN_EMP;
-- Create the GEN_EMP table with a row_id column
CREATE TABLE GEN_EMP AS
SELECT ROWNUM AS row_id, employeeNumber
FROM (
    SELECT DISTINCT employeeNumber
    FROM employees
);

-- view the contents of the GEN_EMP table
SELECT count(*) as "GEN_EMP contents" FROM GEN_EMP;
```

```
-- (iii)To generate > 10,000 records in the temp_emp table with
unique
```

```
employeeNumber values starting from 500 and incrementing by 1,
DROP TABLE temp_emp;
DROP TABLE temp_emp_unique;
-- Create a new table to hold the unique employee data
CREATE TABLE temp_emp AS SELECT * FROM employees;
CREATE TABLE temp_emp_unique AS SELECT * FROM employees WHERE 1=0;

-- Create a sequence to generate unique employee numbers starting
from 1703
drop sequence emp_number_sequence;
CREATE SEQUENCE emp_number_sequence
  START WITH 1703
  INCREMENT BY 1;

-- Loop to insert 10,000 records into the temp_emp table with
unique customer numbers
DECLARE
-- Initialize v_record_count
  v_counter NUMBER := 0;
BEGIN
  WHILE v_counter < 500 LOOP
    INSERT INTO temp_emp (employeeNumber, lastName, firstName,
extension, email, officeCode, reportsTo, jobTitle)
    SELECT emp_number_sequence.NEXTVAL, lastName, firstName,
extension, email, officeCode, reportsTo, jobTitle
    FROM employees;

    v_counter := v_counter + 1;
  END LOOP;
END;
/
-- Append the generated records at the end of temp_emp
INSERT INTO temp_emp
SELECT * FROM temp_emp_unique;
SELECT COUNT(*) as "Records in temp_emp" FROM temp_emp;
-- 11500
```

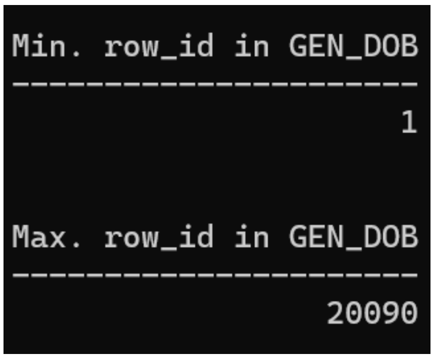
```
--(iv) Assign the random DOB, gender to new_emp,
--Generate updated records for some employee and assign random
endDate(must be > startDate), assign the old record’s ‘status’ as
del
```

```
DROP TABLE new_emp;
CREATE TABLE new_emp AS SELECT * FROM employee_dim WHERE 1=0;

drop sequence emp_sequence;
create sequence emp_sequence
start with 100001
increment by 1;
DECLARE
  CURSOR emp_cur IS
    SELECT *
    FROM temp_emp
    ORDER BY employeeNumber;

  t_rec emp_cur%ROWTYPE;

  v_random_rowID NUMBER;
  v_dob DATE;
  v_random_reportsTo  NUMBER;

  v_gender CHAR(1);
  v_startDate DATE;
  v_endDate DATE;
  v_status VARCHAR2(3); -- Assuming status is a VARCHAR2(3) field

  -- Variables for controlling the percentage of records with
endDate
  v_records_count NUMBER := 0;
  v_records_with_endDate NUMBER;
  v_percentage NUMBER := 15; -- Change this to set the desired
percentage
  v_counter NUMBER := 0; -- Counter variable to keep track of the
row number in new_emp

BEGIN
  -- Get the total number of records
  SELECT COUNT(*) as "Records in temp_emp" INTO v_records_count
FROM temp_emp;

  -- Calculate the number of records with endDate
  v_records_with_endDate := ROUND(v_records_count * v_percentage /
100);

  FOR t_rec IN emp_cur LOOP

    -- Assign DOB randomly
    v_random_rowID := TRUNC(DBMS_RANDOM.VALUE(1, 20091));
    SELECT dob_date INTO v_dob
    FROM Gen_DOB
    WHERE row_id = v_random_rowID;

    -- Assign gender randomly
    IF (MOD(v_random_rowID, 2) = 0) THEN
      v_gender := 'M';
    ELSE
      v_gender := 'F';
    END IF;
    -- Generate a random number of days to add to the start date
    v_startDate := TO_DATE('01/01/2020', 'DD/MM/YYYY') +
TRUNC(DBMS_RANDOM.VALUE(1, TO_DATE('01/01/2022', 'DD/MM/YYYY') -
TO_DATE('01/01/2000', 'DD/MM/YYYY')));

    -- Determine if this record should have an endDate
    IF v_records_with_endDate > 0 AND v_counter > 23  AND
MOD(t_rec.employeeNumber, 5) = 0 THEN
      -- Generate a random endDate greater than startDate
      v_endDate := v_startDate + TRUNC(DBMS_RANDOM.VALUE(1, 365));
-- Adjust the range as needed
      v_records_with_endDate := v_records_with_endDate - 1;

      -- Assign status as 'DEL' for records with endDate
      v_status := 'DEL';
    ELSE
      v_endDate := NULL; -- No endDate for this record
      v_status := 'ACT';  -- Status is active for records without
endDate
    END IF;
    v_counter := v_counter + 1;

    -- Insert into new_emp
    INSERT INTO new_emp (
      employeeKey,
      employeeNumber,
      lastName,
      firstName,
      extension,
      email,
      officeCode,
      reportsTo,
      jobTitle,
      dob,
      gender,
      startDate,
      endDate,
      status
    )

    VALUES (
      emp_sequence.NEXTVAL,
      t_rec.employeeNumber,
      t_rec.lastName,
      t_rec.firstName,
      t_rec.extension,
      t_rec.email,
      t_rec.officeCode,
      t_rec.reportsTo,
      t_rec.jobTitle,
      v_dob,
      v_gender,
      v_startDate,
      v_endDate,
      v_status
    );
  END LOOP;
END;
/
```

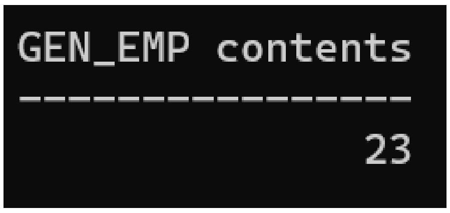
```
SELECT COUNT(*) as "Status = DEL(new_emp)"
FROM new_emp
WHERE status = 'DEL';
-- 15% * 11,500 = 1725
```

```
--(v) Generate random reportsTo for employee with status 'DEL'
```

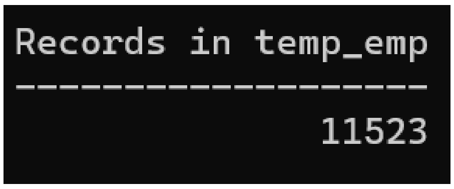
```
-- Create the update_emp table if it doesn't already exist
DROP TABLE update_emp;
CREATE TABLE update_emp AS
SELECT *
FROM new_emp
WHERE status = 'DEL';

-- Update the status, endDate, and reportsTo for all rows in the
updated_emp table
UPDATE update_emp
SET status = 'ACT',
    endDate = NULL,
    reportsTo = (
      SELECT employeeNumber
      FROM GEN_EMP
      WHERE row_id = TRUNC(DBMS_RANDOM.VALUE(1, 24))
      AND ROWNUM = 1
    );

-- Insert all rows from update_emp into new_emp with
emp_sequence.NEXTVAL as customerKey
INSERT INTO new_emp (
    employeeKey,
    employeeNumber,
    lastName,
    firstName,
    extension,
    email,
    officeCode,
    reportsTo,
    jobTitle,
    dob,
    gender,
    startDate,
    endDate,
    status
)
SELECT
  emp_sequence.NEXTVAL,
  employeeNumber,
  lastName,
  firstName,
  extension,
  email,
  officeCode,
  reportsTo,
  jobTitle,
  dob,
  gender,
  startDate,
  endDate,
  status
FROM update_emp;
```



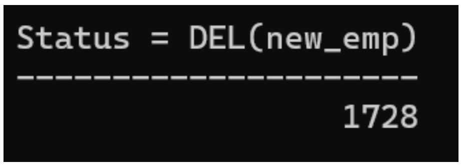
```
Min. row_id in GEN_DOB
----------------------
                     1


Max. row_id in GEN_DOB
----------------------
                 20090
```



```
GEN_EMP contents
----------------
              23
```

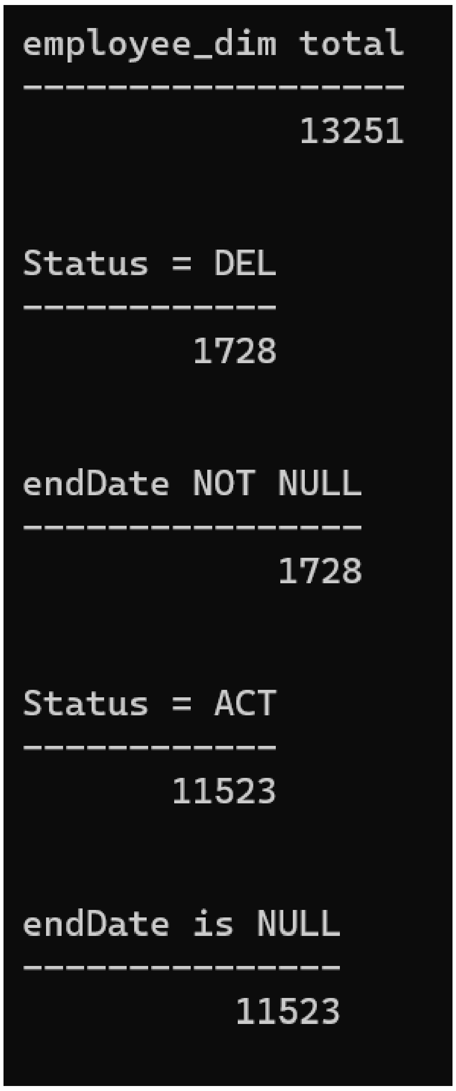


```
Records in temp_emp
-------------------
              11523
```



```
Status = DEL(new_emp)
---------------------
                 1728
```

11523*15% = 1728


```
employee_dim total
------------------
             13251


Status = DEL
------------
        1728


endDate NOT NULL
----------------
            1728


Status = ACT
------------
       11523


endDate is NULL
---------------
          11523
```

2.1.4 For product_dim

```
DROP TABLE product_dim;
CREATE TABLE product_dim (
    productKey          NUMBER(11) NOT NULL,
    productCode         VARCHAR(15) NOT NULL,
    productName         VARCHAR(70) NOT NULL,
    productLine         VARCHAR(50) NOT NULL,
    productScale        VARCHAR(10) NOT NULL,
    productVendor       VARCHAR(50) NOT NULL,
    productDescription  VARCHAR(4000) NOT NULL,
    quantityInStock     NUMBER(4) NOT NULL,
    buyPrice            NUMBER(7,2) NOT NULL,
    MSRP                NUMBER(7,2) NOT NULL,
    PRIMARY KEY (productKey)
);

-- (i) Copy data in products to product_dim
 drop sequence product_seq;
 create sequence product_seq
 start with 100001
 increment by 1;
delete product_dim;
-- ETL for product_dim
 insert into product_dim
 select product_seq.nextval,
        productCode,
        upper(substr(productName,1,70)),
        upper(substr(productLine,1,50)),
        productScale,
        upper(substr(productVendor,1,50)),
        upper(substr(productDescription,1,4000)),
        quantityInStock,
        buyPrice,
        MSRP
 from products;

select count(*) as "product_dim total"
from product_dim;
--110 rows
COMMIT;
```

2.1.5 For office_dim

```
-- (i)To generate 350 records in the temp_office_unique table with
unique officeNumber values starting from 500 and incrementing by 1
```

```
DROP TABLE office_dim;
CREATE TABLE office_dim (
    officeKey       NUMBER(11) NOT NULL,
    officeCode         VARCHAR(10) NOT NULL,
    city               VARCHAR(50) NOT NULL,
    phone              VARCHAR(50) NOT NULL,
    addressLine1   VARCHAR(50) NOT NULL,
    addressLine2   VARCHAR(50) DEFAULT NULL,
    state              VARCHAR(50) DEFAULT NULL,
    country            VARCHAR(50) NOT NULL,
    postalCode         VARCHAR(15) NOT NULL,
    territory    VARCHAR(10) NOT NULL,
    PRIMARY KEY (officeKey)
);

DROP TABLE temp_office;
DROP TABLE temp_office_unique;
-- Create a new table to hold the unique office data
CREATE TABLE temp_office AS SELECT * FROM offices;
CREATE TABLE temp_office_unique AS SELECT * FROM offices WHERE 1=0;


-- Create a sequence to generate unique office numbers starting
from 500
drop sequence office_number_sequence;
CREATE SEQUENCE office_number_sequence
  START WITH 8
  INCREMENT BY 1;

-- Loop to insert 350 records into the temp_office table with
```

```
unique office numbers
DECLARE
-- Initialize v_record_count
  v_counter NUMBER := 0;
BEGIN
--  WHILE v_counter < 50 LOOP
    WHILE v_counter < 1 LOOP
    INSERT INTO temp_office (officeCode, city, phone, addressLine1,
addressLine2, state, country, postalCode, territory)
    SELECT office_number_sequence.NEXTVAL, city, phone,
addressLine1, addressLine2, state, country, postalCode, territory
    FROM offices;

    v_counter := v_counter + 1;
  END LOOP;
END;
/
-- Append the generated records at the end of temp_office
INSERT INTO temp_office
SELECT * FROM temp_office_unique;

SELECT COUNT(*) as "Records in temp_office" FROM temp_office;
--  350
```

```
--(ii) Assign the random state & city to new_office
```

```
DROP TABLE new_office;
CREATE TABLE new_office AS SELECT * FROM office_dim WHERE 1=0;

drop sequence office_sequence;
create sequence office_sequence
start with 100001
increment by 1;

DECLARE
  CURSOR office_cur IS
    SELECT *
    FROM temp_office
    ORDER BY officeCode;

  t_rec office_cur%ROWTYPE;

  v_city VARCHAR(30);
  v_state VARCHAR(30);
  v_ID NUMBER;


BEGIN
  FOR t_rec IN office_cur LOOP
    -- Assign state and city randomly
    -- Generate a random v_ID within a valid range
    v_ID := TRUNC(DBMS_RANDOM.VALUE(10001, 10042));

    -- Query the StateAndCity table to get city and state for the
generated v_ID
```

```
    SELECT city, state INTO v_city, v_state
    FROM StateAndCity
    WHERE StateAndCityID = v_ID;


    -- Insert into new_office
    INSERT INTO new_office (
        officeKey,
        officeCode,
        city,
        phone,
        addressLine1,
        addressLine2,
        state,
        country,
        postalCode,
        territory
    )
    VALUES (
        office_sequence.NEXTVAL,
        t_rec.officeCode,
        v_city,
        t_rec.phone,
        t_rec.addressLine1,
        t_rec.addressLine2,
        v_state,
        t_rec.country,
        t_rec.postalCode,
        t_rec.territory
    );
  END LOOP;
END;
/

SELECT COUNT(*) as "Records in new_office"
FROM new_office;
```

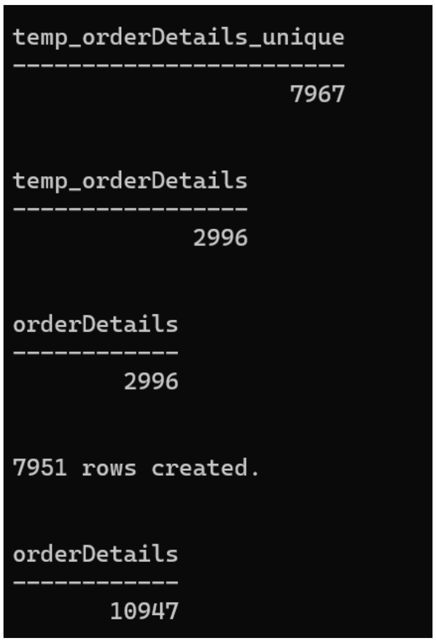
2.1.6 For sales_fact

To generate 100K orders in orders table

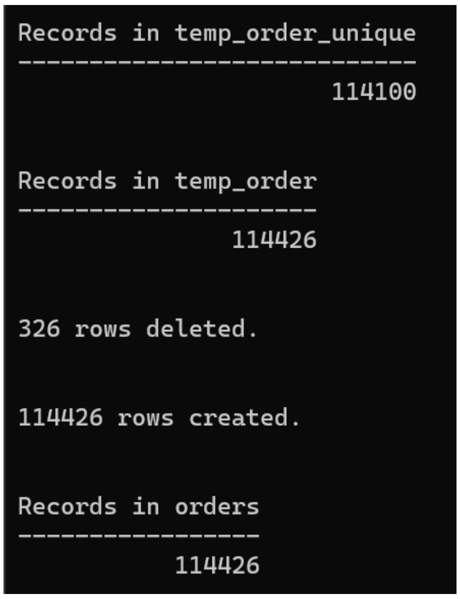
```
DROP TABLE temp_order;
DROP TABLE orderdetails;
DROP TABLE temp_order_unique;
DROP TABLE cust_no_list;
-- Create a new table to hold the unique customer data
CREATE TABLE temp_order AS SELECT * FROM orders;
CREATE TABLE temp_order_unique AS SELECT * FROM orders WHERE 1=0;
-- Create the cust_no_list table if it doesn't already exist
CREATE TABLE cust_no_list (
  row_id NUMBER,
  customerNumber NUMBER(11)
);

-- Insert unique customerNumbers from customer_dim into
cust_no_list
INSERT INTO cust_no_list (row_id, customerNumber)
```

```
SELECT ROWNUM, customerNumber
FROM (
  SELECT DISTINCT customerNumber
  FROM customer_dim
);

-- Create a sequence to generate unique customer numbers starting
from 500
drop sequence order_number_sequence;

CREATE SEQUENCE order_number_sequence
  START WITH 10426
  INCREMENT BY 1;

-- Loop to insert 10,000 records into the temp_order table with
unique customer numbers
DECLARE
-- Initialize v_record_count
  v_counter NUMBER := 0;
  v_cust_no NUMBER(11);
  v_random_rowID NUMBER;
BEGIN
  WHILE v_counter < 35 LOOP
    -- Assign a random row_id
    v_random_rowID := TRUNC(DBMS_RANDOM.VALUE(1, 12323));

    -- Get the customerNumber based on the random row_id
    SELECT customerNumber INTO v_cust_no
    FROM cust_no_list
    WHERE row_id = v_random_rowID
    AND customerNumber IS NOT NULL; -- Ensure non-NULL
customerNumber

    INSERT INTO temp_order_unique
(orderNumber,orderDate,requiredDate,shippedDate,status,comments,cus
tomerNumber)
    SELECT
order_number_sequence.NEXTVAL,orderDate,requiredDate,shippedDate,st
atus,comments,customerNumber
    FROM orders;

    v_counter := v_counter + 1;
  END LOOP;
END;
/

SELECT COUNT(*) as "Records in temp_order_unique" FROM
temp_order_unique;

-- Append the generated records at the end of temp_order
INSERT INTO temp_order
SELECT * FROM temp_order_unique;

-- Assign orderDate randomly

--Generate orderDate
drop table Gen_Date;

create table Gen_Date
(row_id number,
orderDate date);

set serveroutput on
create or replace procedure proc_gen_date(in_start_Date IN varchar,
                   in_end_Date IN varchar) IS
  v_startDate date;
  v_endDate   date;
  v_counter number :=1; --so that row_id is starting from 1

begin
   v_startDate:=to_date(in_start_Date,'dd/mm/yyyy');
   v_endDate  :=to_date(in_end_Date,'dd/mm/yyyy');

   while (v_startDate<=v_endDate) loop
       insert into Gen_Date values(v_counter,v_startDate);
       v_counter:=v_counter+1;
       v_startDate:= v_startDate+1;

   end loop;
dbms_output.put_line('Count is ' ||v_counter);
end;
/
exec proc_gen_date('06/01/2021','31/05/2023');
--Count is 877

--update all the orderDate in temp_order record, random select from
the Gen_Date
DECLARE
    v_min_row_id NUMBER;
    v_max_row_id NUMBER;
    v_random_row_id NUMBER;
    v_order_date DATE;
BEGIN
    -- Get the minimum and maximum row IDs from Gen_Date
    SELECT MIN(row_id), MAX(row_id) INTO v_min_row_id, v_max_row_id
FROM Gen_Date;

    -- Loop through each row in temp_order
    FOR rec IN (SELECT rowid, orderDate FROM temp_order) LOOP
        -- Generate a random row ID between v_min_row_id and
v_max_row_id
        v_random_row_id := TRUNC(DBMS_RANDOM.VALUE(v_min_row_id,
v_max_row_id));

        -- Get the corresponding orderDate from Gen_Date
        SELECT orderDate INTO v_order_date FROM Gen_Date WHERE
row_id = v_random_row_id;

        -- Update the orderDate for the current row in temp_order
        UPDATE temp_order
        SET orderDate = v_order_date
        WHERE rowid = rec.rowid;
    END LOOP;

    COMMIT;
END;
/

SELECT COUNT(*) as "Records in temp_order_unique" FROM
temp_order_unique;
```

```
SELECT COUNT(*) as "Records in temp_order" FROM temp_order;
delete orders;
--Insert temp_order into orders
INSERT INTO orders (orderNumber, orderDate, requiredDate,
shippedDate, status, comments, customerNumber)
SELECT orderNumber, orderDate, requiredDate, shippedDate, status,
comments, customerNumber
FROM temp_order;

SELECT COUNT(*) as "Records in orders" FROM orders;
-- 11736
```



```
temp_orderDetails_unique
------------------------
                    7967


temp_orderDetails
-----------------
             2996


orderDetails
------------
        2996


7951 rows created.


orderDetails
------------
       10947
```

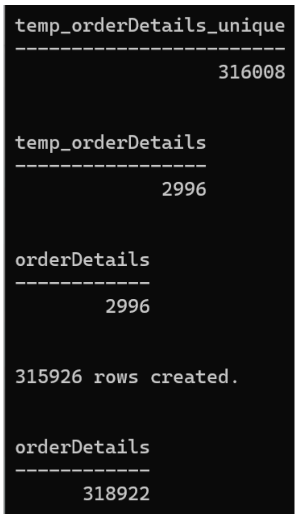


```
Records in temp_order_unique
----------------------------
                      114100


Records in temp_order
---------------------
               114426


326 rows deleted.


114426 rows created.


Records in orders
-----------------
           114426
```

```
To generate 300K orderDetails
```

```
-- Drop existing temporary tables if they exist
DROP TABLE temp_orderDetails;
DROP TABLE temp_orderDetails_unique;
DROP TABLE order_no_list;

-- Create a new table to hold the unique orderNumber values
CREATE TABLE temp_orderDetails AS SELECT * FROM orderDetails;
CREATE TABLE temp_orderDetails_unique AS SELECT * FROM orderDetails
WHERE 1=0;
-- Create the order_no_list table if it doesn't already exist
CREATE TABLE order_no_list (
  row_id NUMBER,
  orderNumber NUMBER(11)
);

-- Insert unique orderNumbers from orders into order_no_list
INSERT INTO order_no_list (row_id, orderNumber)
SELECT ROWNUM, orderNumber
FROM (
  SELECT DISTINCT orderNumber
  FROM orders
);
DECLARE
  v_random_rowID NUMBER;
  v_order_no NUMBER;
  v_product_id NUMBER;
  v_product_code VARCHAR2(15);
  v_quantity NUMBER;
  v_price NUMBER;
  v_order_line NUMBER;
  v_counter NUMBER := 0;
BEGIN
  WHILE v_counter < 8000 LOOP
    -- Generate a random product_id between 100001 and the maximum
productKey
    SELECT TRUNC(DBMS_RANDOM.VALUE(100001, (SELECT MAX(productKey)
FROM product_dim)))
    INTO v_product_id
    FROM DUAL;

    -- Generate a random row_id between 1 and 11736 (MAX(ROW_ID)
from order_no_list)
    v_random_rowID := TRUNC(DBMS_RANDOM.VALUE(1, 11737));

    -- Get the orderNumber based on the random row_id
    SELECT orderNumber INTO v_order_no
    FROM order_no_list
    WHERE row_id = v_random_rowID;

    -- Get the productCode based on the random product_id
    SELECT productCode INTO v_product_code
    FROM product_dim
    WHERE productKey = v_product_id;

    -- Generate random values for quantityOrdered, priceEach, and
orderLineNumber
    v_quantity := TRUNC(DBMS_RANDOM.VALUE(1, 10)); -- Adjust the
range as needed
    v_price := ROUND(DBMS_RANDOM.VALUE(10, 1000), 2); -- Adjust the
range as needed
```

```
    v_order_line := TRUNC(DBMS_RANDOM.VALUE(1, 10)); -- Adjust the
range as needed

    -- Insert the record into the temp_orderDetails_unique table if
the combination is unique
    INSERT INTO temp_orderDetails_unique (orderNumber, productCode,
quantityOrdered, priceEach, orderLineNumber)
    SELECT v_order_no, v_product_code, v_quantity, v_price,
v_order_line
    FROM DUAL
    WHERE NOT EXISTS (
      SELECT 1
      FROM temp_orderDetails_unique t
      WHERE t.orderNumber = v_order_no AND t.productCode =
v_product_code
    );

    v_counter := v_counter + 1;
  END LOOP;
END;
/

select count(*) as "temp_orderDetails_unique" from
temp_orderDetails_unique;
select count(*) as "temp_orderDetails" from temp_orderDetails;

-- Append the records in temp_orderDetails_unique at the end of
orderDetails

select count(*) as "orderDetails" from orderDetails;
-- Append unique records from temp_orderDetails_unique into
orderDetails
INSERT INTO orderDetails (orderNumber, productCode,
quantityOrdered, priceEach, orderLineNumber)
SELECT t.orderNumber, t.productCode, t.quantityOrdered,
t.priceEach, t.orderLineNumber
FROM temp_orderDetails_unique t
WHERE NOT EXISTS (
  SELECT 1
  FROM orderDetails o
  WHERE o.orderNumber = t.orderNumber
  AND o.productCode = t.productCode
);

select count(*) as "orderDetails" from orderDetails;
--318922
```



```
temp_orderDetails_unique
------------------------
                  316008


temp_orderDetails
-----------------
             2996


orderDetails
------------
        2996


315926 rows created.


orderDetails
------------
      318922
```

2.2 Script for subsequent loading

2.2.1 date_dim

```
(ii)Assign Malaysia holiday
```

```
--In 2021
-- Update HOLIDAY_IND for 2021 based on provided holiday dates
UPDATE date_dim
SET HOLIDAY_IND = 'Y'
WHERE year = 2021
AND cal_date IN (
    TO_DATE('2021-01-01', 'YYYY-MM-DD'),  -- New Year's Day
    TO_DATE('2021-01-28', 'YYYY-MM-DD'),  -- Thaipusam
    TO_DATE('2021-02-12', 'YYYY-MM-DD'),  -- CNY
    TO_DATE('2021-02-13', 'YYYY-MM-DD'),  -- CNY
    TO_DATE('2021-02-14', 'YYYY-MM-DD'),  -- CNY
    TO_DATE('2021-04-13', 'YYYY-MM-DD'),  -- Awal Ramadan
    TO_DATE('2021-04-29', 'YYYY-MM-DD'),  -- Nuzul Al-Quran
    TO_DATE('2021-05-01', 'YYYY-MM-DD'),  -- Labour Day
    TO_DATE('2021-05-13', 'YYYY-MM-DD'),  -- Hari Raya Aidilfitri
    TO_DATE('2021-05-14', 'YYYY-MM-DD'),  -- Hari Raya Aidilfitri
Holiday
    TO_DATE('2021-05-26', 'YYYY-MM-DD'),  -- Wesak Day
    TO_DATE('2021-06-07', 'YYYY-MM-DD'),  -- Agong's Birthday
    TO_DATE('2021-07-20', 'YYYY-MM-DD'),  -- Hari Raya Haji
    TO_DATE('2021-08-10', 'YYYY-MM-DD'),  -- Awal Muharram
    TO_DATE('2021-08-31', 'YYYY-MM-DD'),  -- Merdeka Day
    TO_DATE('2021-09-16', 'YYYY-MM-DD'),  -- Malaysia Day
    TO_DATE('2021-10-19', 'YYYY-MM-DD'),  -- Prophet Muhammad's
Birthday
    TO_DATE('2021-12-25', 'YYYY-MM-DD'),  -- Christmas Day
    TO_DATE('2021-12-26', 'YYYY-MM-DD')   -- Christmas Day
);
```

```
-- Update HOLIDAY_IND for 2022 based on provided holiday dates
UPDATE date_dim
SET HOLIDAY_IND = 'Y'
WHERE year = 2022
AND cal_date IN (
    TO_DATE('2022-01-01', 'YYYY-MM-DD'),  -- New Year's Day
    TO_DATE('2022-01-18', 'YYYY-MM-DD'),  -- Thaipusam
    TO_DATE('2022-02-01', 'YYYY-MM-DD'),  -- Chinese New Year
    TO_DATE('2022-02-02', 'YYYY-MM-DD'),  -- Chinese New Year
Holiday
    TO_DATE('2022-04-19', 'YYYY-MM-DD'),  -- Nuzul Al-Quran
    TO_DATE('2022-05-01', 'YYYY-MM-DD'),  -- Labour Day
    TO_DATE('2022-05-02', 'YYYY-MM-DD'),  -- Hari Raya Aidilfitri
    TO_DATE('2022-05-03', 'YYYY-MM-DD'),  -- Hari Raya Aidilfitri
Holiday
    TO_DATE('2022-05-04', 'YYYY-MM-DD'),  -- Labour Day Holiday
    TO_DATE('2022-05-15', 'YYYY-MM-DD'),  -- Wesak Day
    TO_DATE('2022-05-16', 'YYYY-MM-DD'),  -- Wesak Day Holiday
    TO_DATE('2022-06-06', 'YYYY-MM-DD'),  -- Agong's Birthday
    TO_DATE('2022-07-10', 'YYYY-MM-DD'),  -- Hari Raya Haji
    TO_DATE('2022-08-31', 'YYYY-MM-DD'),  -- Merdeka Day
    TO_DATE('2022-09-16', 'YYYY-MM-DD'),  -- Malaysia Day
    TO_DATE('2022-10-09', 'YYYY-MM-DD'),  -- Prophet Muhammad's
Birthday
    TO_DATE('2022-10-10', 'YYYY-MM-DD'),  -- Prophet Muhammad's
Birthday Holiday
    TO_DATE('2022-11-18', 'YYYY-MM-DD'),  -- Special Public Holiday
(GE15)
    TO_DATE('2022-11-19', 'YYYY-MM-DD'),  -- Special Public Holiday
(GE15)
    TO_DATE('2022-11-28', 'YYYY-MM-DD'),  -- Special Public Holiday
28 Nov
    TO_DATE('2022-12-25', 'YYYY-MM-DD'),  -- Christmas Day
    TO_DATE('2022-12-26', 'YYYY-MM-DD')   -- Christmas Holiday
);
```

```
-- Update HOLIDAY_IND for 2023 based on provided holiday dates
UPDATE date_dim
SET HOLIDAY_IND = 'Y'
WHERE year = 2023
AND cal_date IN (
    TO_DATE('2023-01-01', 'YYYY-MM-DD'),  -- New Year's Day
    TO_DATE('2023-01-02', 'YYYY-MM-DD'),  -- New Year Holiday
    TO_DATE('2023-01-22', 'YYYY-MM-DD'),  -- Chinese New Year
    TO_DATE('2023-01-23', 'YYYY-MM-DD'),  -- Chinese New Year
Holiday
    TO_DATE('2023-01-24', 'YYYY-MM-DD'),  -- Chinese New Year
Holiday
    TO_DATE('2023-02-06', 'YYYY-MM-DD'),  -- Thaipusam Holiday
    TO_DATE('2023-04-08', 'YYYY-MM-DD'),  -- Nuzul Al-Quran
    TO_DATE('2023-04-21', 'YYYY-MM-DD'),  -- Hari Raya Aidilfitri
Holiday
    TO_DATE('2023-04-22', 'YYYY-MM-DD'),  -- Hari Raya Aidilfitri
    TO_DATE('2023-04-23', 'YYYY-MM-DD'),  -- Hari Raya Aidilfitri
Holiday
```

```
    TO_DATE('2023-04-24', 'YYYY-MM-DD'),  -- Hari Raya Aidilfitri
Holiday
    TO_DATE('2023-04-26', 'YYYY-MM-DD'),  -- Sultan of Terengganu's
Birthday
    TO_DATE('2023-05-01', 'YYYY-MM-DD'),  -- Labour Day
    TO_DATE('2023-05-04', 'YYYY-MM-DD')   -- Wesak Day
);
```

(iii)Assign Malaysia festive code

```
-- Update festive_code for 2021 based on provided holiday dates
UPDATE date_dim
SET festive_code =
    CASE
        WHEN cal_date = TO_DATE('2021-01-01', 'YYYY-MM-DD') THEN
'New Year''s Day'
        WHEN cal_date = TO_DATE('2021-01-28', 'YYYY-MM-DD') THEN
'Thaipusam'
        WHEN cal_date = TO_DATE('2021-02-12', 'YYYY-MM-DD') THEN
'CNY'
        WHEN cal_date = TO_DATE('2021-02-13', 'YYYY-MM-DD') THEN
'CNY'
        WHEN cal_date = TO_DATE('2021-02-14', 'YYYY-MM-DD') THEN
'CNY'
        WHEN cal_date = TO_DATE('2021-04-13', 'YYYY-MM-DD') THEN
'Awal Ramadan'
        WHEN cal_date = TO_DATE('2021-04-29', 'YYYY-MM-DD') THEN
'Nuzul Al-Quran'
        WHEN cal_date = TO_DATE('2021-05-01', 'YYYY-MM-DD') THEN
'Labour Day'
        WHEN cal_date = TO_DATE('2021-05-13', 'YYYY-MM-DD') THEN
'Hari Raya Aidilfitri'
        WHEN cal_date = TO_DATE('2021-05-14', 'YYYY-MM-DD') THEN
'Hari Raya Aidilfitri Holiday'
        WHEN cal_date = TO_DATE('2021-05-26', 'YYYY-MM-DD') THEN
'Wesak Day'
        WHEN cal_date = TO_DATE('2021-06-07', 'YYYY-MM-DD') THEN
'Agong''s Birthday'
        WHEN cal_date = TO_DATE('2021-07-20', 'YYYY-MM-DD') THEN
'Hari Raya Haji'
        WHEN cal_date = TO_DATE('2021-08-10', 'YYYY-MM-DD') THEN
'Awal Muharram'
        WHEN cal_date = TO_DATE('2021-08-31', 'YYYY-MM-DD') THEN
'Merdeka Day'
        WHEN cal_date = TO_DATE('2021-09-16', 'YYYY-MM-DD') THEN
'Malaysia Day'
        WHEN cal_date = TO_DATE('2021-10-19', 'YYYY-MM-DD') THEN
'Prophet Muhammad''s Birthday'
        WHEN cal_date = TO_DATE('2021-12-25', 'YYYY-MM-DD') THEN
'Christmas Day'
        WHEN cal_date = TO_DATE('2021-12-26', 'YYYY-MM-DD') THEN
'Christmas Day'
        ELSE NULL -- Set to NULL for non-holiday dates
    END
WHERE year = 2021;
```

```
-- Update festive_code for 2022 based on provided holiday dates
UPDATE date_dim
```

```
SET festive_code =
    CASE
        WHEN cal_date = TO_DATE('2022-01-01', 'YYYY-MM-DD') THEN
'NY'
        WHEN cal_date = TO_DATE('2022-01-18', 'YYYY-MM-DD') THEN
'Thaipusam'
        WHEN cal_date = TO_DATE('2022-02-01', 'YYYY-MM-DD') THEN
'CNT'
        WHEN cal_date = TO_DATE('2022-02-02', 'YYYY-MM-DD') THEN
'CNY'
        WHEN cal_date = TO_DATE('2022-04-19', 'YYYY-MM-DD') THEN
'Al-Quran'
        WHEN cal_date = TO_DATE('2022-05-01', 'YYYY-MM-DD') THEN
'Labour'
        WHEN cal_date = TO_DATE('2022-05-02', 'YYYY-MM-DD') THEN
'Raya'
        WHEN cal_date = TO_DATE('2022-05-03', 'YYYY-MM-DD') THEN
'Raya'
        WHEN cal_date = TO_DATE('2022-05-04', 'YYYY-MM-DD') THEN
'Labour'
        WHEN cal_date = TO_DATE('2022-05-15', 'YYYY-MM-DD') THEN
'Wesak'
        WHEN cal_date = TO_DATE('2022-05-16', 'YYYY-MM-DD') THEN
'Wesak Day'
        WHEN cal_date = TO_DATE('2022-06-06', 'YYYY-MM-DD') THEN
'Agong'
        WHEN cal_date = TO_DATE('2022-07-10', 'YYYY-MM-DD') THEN
'Haji'
        WHEN cal_date = TO_DATE('2022-08-31', 'YYYY-MM-DD') THEN
'Merdeka'
        WHEN cal_date = TO_DATE('2022-09-16', 'YYYY-MM-DD') THEN
'Malaysia'
        WHEN cal_date = TO_DATE('2022-10-09', 'YYYY-MM-DD') THEN
'Prophet Muhammad'
        WHEN cal_date = TO_DATE('2022-10-10', 'YYYY-MM-DD') THEN
'Prophet Muhammad'
        WHEN cal_date = TO_DATE('2022-11-18', 'YYYY-MM-DD') THEN
'GE15'
        WHEN cal_date = TO_DATE('2022-11-19', 'YYYY-MM-DD') THEN
'GE15'
        WHEN cal_date = TO_DATE('2022-11-28', 'YYYY-MM-DD') THEN
'GE15'
        WHEN cal_date = TO_DATE('2022-12-25', 'YYYY-MM-DD') THEN
'XMAS'
        WHEN cal_date = TO_DATE('2022-12-26', 'YYYY-MM-DD') THEN
'XMAS'
        ELSE NULL -- Set to NULL for non-holiday dates
    END
WHERE year = 2022;
```

```
-- Update festive_code for 2023 based on provided holiday dates
UPDATE date_dim
SET festive_code =
    CASE
        WHEN cal_date = TO_DATE('2023-01-01', 'YYYY-MM-DD') THEN
'NY'
        WHEN cal_date = TO_DATE('2023-01-02', 'YYYY-MM-DD') THEN
'NY'
```

```
        WHEN cal_date = TO_DATE('2023-01-22', 'YYYY-MM-DD') THEN
'CNY'
        WHEN cal_date = TO_DATE('2023-01-23', 'YYYY-MM-DD') THEN
'CNY'
        WHEN cal_date = TO_DATE('2023-01-24', 'YYYY-MM-DD') THEN
'CNY'
        WHEN cal_date = TO_DATE('2023-02-06', 'YYYY-MM-DD') THEN
'Thaipusam '
        WHEN cal_date = TO_DATE('2023-04-08', 'YYYY-MM-DD') THEN '
Al-Quran'
        WHEN cal_date = TO_DATE('2023-04-21', 'YYYY-MM-DD') THEN
'RAYA'
        WHEN cal_date = TO_DATE('2023-04-22', 'YYYY-MM-DD') THEN
'RAYA'
        WHEN cal_date = TO_DATE('2023-04-23', 'YYYY-MM-DD') THEN
'RAYA'
        WHEN cal_date = TO_DATE('2023-04-24', 'YYYY-MM-DD') THEN
'RAYA'
        WHEN cal_date = TO_DATE('2023-05-01', 'YYYY-MM-DD') THEN
'Labour'
        WHEN cal_date = TO_DATE('2023-05-04', 'YYYY-MM-DD') THEN
'Wesak'
        ELSE NULL -- Set to NULL for non-holiday dates
    END
WHERE year = 2023;
```

2.2.2 customer_dim

```
-- (vi) Copy data in new_cust to customer_dim

drop sequence cust_seq;
 create sequence cust_seq
 start with 100001
 increment by 1;

delete customer_dim;
-- ETL for customer_dim
 insert into customer_dim
 select cust_seq.nextval,
        customerNumber,
        upper(substr(customerName,1,50)),
        upper(substr(contactLastName,1,50)),
        upper(substr(contactFirstName,1,50)),
        phone,
        upper(substr(addressLine1,1,50)),
        upper(substr(addressLine2,1,50)),
        upper(city),
        upper(state),
        postalCode,
        upper(country),
        salesRepEmployeeNumber,
        creditLimit,
        dob,
        upper(gender),
        startDate,
        endDate,
        status
 from new_cust;
-- (iv) Update all country into Malaysia
UPDATE customer_dim
SET country = 'Malaysia';

SELECT COUNT(*) as "Customer_dim total" FROM customer_dim;
--Old record: With an endDate and status is 'DEL'
SELECT COUNT(*) as "Status = DEL"
FROM customer_dim
WHERE status = 'DEL';
--14780ge
SELECT COUNT(*) as "endDate NOT NULL"
FROM customer_dim
WHERE endDate IS NOT NULL;

--Active record: Without endDate and status is 'ACT'
SELECT COUNT(*) as "Status = ACT"
FROM customer_dim
WHERE status = 'ACT';

SELECT COUNT(*) as "endDate is NULL"
FROM customer_dim
WHERE endDate IS NULL;

COMMIT;
```

2.2.3 employee_dim

```
-- (vi) Copy data in new_emp to employee_dim
```

```
delete employee_dim;
 drop sequence emp_seq;
 create sequence emp_seq
 start with 100001
 increment by 1;

-- ETL for employee_dim
 insert into employee_dim
 select emp_seq.nextval,
        employeeNumber,
        upper(substr(lastName,1,50)),
        upper(substr(firstName,1,50)),
        upper(extension),
        upper(email),
        officeCode,
        reportsTo,
        jobTitle,
        dob,
        upper(gender),
        startDate,
        endDate,
        status
 from new_emp;

SELECT COUNT(*) as "employee_dim total" FROM employee_dim;
--Old record: With an endDate and status is 'DEL'
SELECT COUNT(*) as "Status = DEL"
FROM employee_dim
WHERE status = 'DEL';

SELECT COUNT(*) as "endDate NOT NULL"
FROM employee_dim
WHERE endDate IS NOT NULL;

--Active record: Without endDate and status is 'ACT'
SELECT COUNT(*) as "Status = ACT"
FROM employee_dim
WHERE status = 'ACT';

SELECT COUNT(*) as "endDate is NULL"
FROM employee_dim
WHERE endDate IS NULL;

COMMIT;
```

2.2.4 product_dim

```
-- Copy data in products to product_dim
```

```
drop sequence product_seq;
 create sequence product_seq
 start with 100001
 increment by 1;
delete product_dim;
-- ETL for product_dim
 insert into product_dim
 select product_seq.nextval,
        productCode,
        upper(substr(productName,1,70)),
        upper(substr(productLine,1,50)),
        productScale,
        upper(substr(productVendor,1,50)),
        upper(substr(productDescription,1,4000)),
        quantityInStock,
        buyPrice,
        MSRP
 from products;

select count(*) as "product_dim total"
from product_dim;
```

```
--110 rows
COMMIT;
```

2.2.5 office_dim

```
-- (iii) Copy data in new_office to office_dim
```

```
delete office_dim;
 drop sequence office_seq;
 create sequence office_seq
 start with 100001
 increment by 1;

-- ETL for office_dim
 insert into office_dim
 select office_seq.nextval,
        officeCode,
        upper(city),
        phone,
        upper(substr(addressLine1,1,50)),
        upper(substr(addressLine2,1,50)),
        upper(state),
        upper(country),
        postalCode,
        upper(territory)
 from new_office;

-- (iv) Update all country into Malaysia
UPDATE office_dim
SET country = 'Malaysia';

select count(*) as "office_dim total"
from office_dim;
--350 rows
COMMIT;
```

2.6 For sales_fact

```
Load data into sales_fact from orders, orderdetails, and other
source tables
```

```
ALTER SESSION SET NLS_DATE_FORMAT='YYYY-MM-DD';

DROP TABLE sales_fact;
CREATE TABLE sales_fact (
    salesKey            NUMBER(11) NOT NULL,
    productKey          NUMBER(11) NOT NULL,
    customerKey         NUMBER(11) NOT NULL,
    dateKey             NUMBER NOT NULL,
    employeeKey         NUMBER(11) NOT NULL,
    officeKey           NUMBER(11) NOT NULL,
    orderStatus         VARCHAR(15) NOT NULL,
    quantityOrdered     NUMBER(4) NOT NULL,
    priceEach           NUMBER(7,2) NOT NULL,
    totalAmount         NUMBER(10,2) NOT NULL,
    PRIMARY KEY (salesKey),
    FOREIGN KEY (productKey) REFERENCES product_dim(productKey),
    FOREIGN KEY (customerKey) REFERENCES customer_dim(customerKey),
    FOREIGN KEY (dateKey) REFERENCES date_dim(date_key),
    FOREIGN KEY (employeeKey) REFERENCES employee_dim(employeeKey),
    FOREIGN KEY (officeKey) REFERENCES office_dim(officeKey)
);


 drop sequence sales_seq;
 create sequence sales_seq
 start with 100001
 increment by 1;
 DELETE sales_fact;

-- ETL for sales_fact
-- Generate a sequence of salesKey values using sales_seq
-- Insert data into sales_fact from orders, orderdetails, and other
source tables
INSERT INTO sales_fact (
    salesKey,
    productKey,
    customerKey,
    dateKey,
    employeeKey,
    officeKey,
    orderStatus,
    quantityOrdered,
    priceEach,
    totalAmount
)
SELECT
sales_seq.nextval,
p.productKey,
c.customerKey,
d.date_key,
e.employeeKey,
f.officeKey,
o.status,
od.quantityOrdered,
od.priceEach,
(od.quantityOrdered * od.priceEach)
FROM orders o,
orderdetails od,
product_dim p,
customer_dim c,
date_dim d,
employee_dim e,
office_dim f
WHERE o.orderNumber = od.orderNumber
AND od.productCode = p.productCode
AND o.customerNumber = c.customerNumber
AND o.orderDate = d.cal_date
AND c.salesRepEmployeeNumber = e.employeeNumber
AND e.officeCode = f.officeCode;

select count(*) as "sales_fact total"
from sales_fact;
--12229

COMMIT;
```

Chapter 3 Business Analytics Reports

3.1 Tang Sharren

3.1.1 Top 10 Employee by Sales Performance

This will prompt the user for a year and quarter(Q1 to Q4), and then display the top 10 employees based on their total sales performance for that specific year and quarter. The output will include their dense rank and percentage of total sales. This "percentage" represents the contribution of each employee to the total sales during the specified time frame.

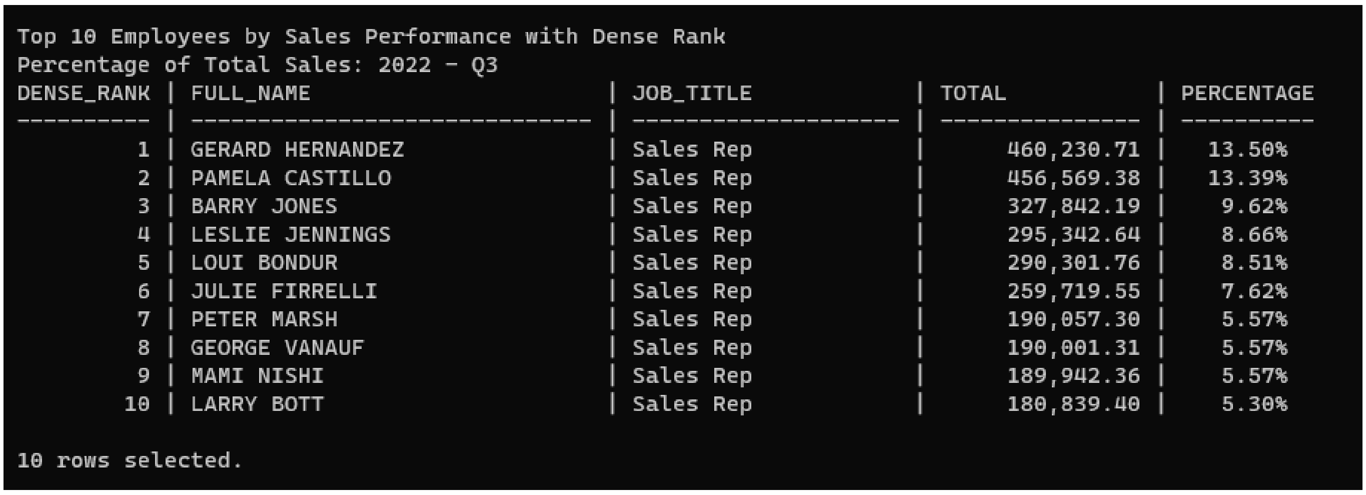
The query helps identify the top-performing employees based on their total sales. This information is valuable for recognizing and rewarding high-achieving employees who contribute significantly to the company's revenue. Also,the company can tailor training and development programs to address specific needs. It helps in optimizing the skill sets of employees. Other than that ,The "percentage of total sales" column indicates each employee's contribution to the total sales for the specified year and quarter. This information can be used to understand the distribution of sales across the salesforce. For employees who do not rank highly, the query provides insights into areas where performance improvement is needed. Managers can provide targeted coaching and support to help employees meet their sales goals.

Script:

```
-- Create a script to set substitution variables
ACCEPT v_cal_year CHAR PROMPT 'Enter the year (2021 - 2023): '
DEFAULT '2023'
ACCEPT v_cal_quarter CHAR PROMPT 'Enter the quarter year (Q1 -
Q4): ' DEFAULT 'Q1'
-- Set a title for the query including the percentage
TTITLE LEFT 'Top 10 Employees by Sales Performance with Dense
Rank' SKIP 1 -
'Percentage of Total Sales: &v_cal_year - &v_cal_quarter' SKIP
1
-- Set SQL*Plus formatting commands
SET SERVEROUTPUT ON
SET PAGESIZE 100
SET LINESIZE 500
SET HEADING ON
SET UNDERLINE ON
SET COLSEP ' | '

-- Create or replace a view to calculate employee sales
performance
-- Create or replace a view to calculate employee sales
performance with DENSE_RANK
CREATE OR REPLACE VIEW Employees_Details AS
SELECT
    TRIM(ED.firstName) || ' ' || TRIM(ED.lastName) AS
full_name,
    TRIM(ED.jobTitle) AS job_title,
    TO_CHAR(SUM(SF.totalAmount), '999,999,999.99') AS total,
    DENSE_RANK() OVER (ORDER BY SUM(SF.totalAmount) DESC) AS
dense_rank,
    TO_CHAR((SUM(SF.totalAmount) / SUM(SUM(SF.totalAmount))
OVER ()) * 100, '999.99') AS percentage
FROM
    sales_fact SF
JOIN
    employee_dim ED ON SF.employeeKey = ED.employeeKey
JOIN
    date_dim DD ON SF.dateKey = DD.date_key
WHERE
    DD.year = TO_NUMBER('&v_cal_year')
    AND DD.quarter = '&v_cal_quarter'
GROUP BY
    TRIM(ED.firstName),
    TRIM(ED.lastName),
    TRIM(ED.jobTitle)
ORDER BY
    dense_rank;
```

```
-- Set column widths for the query output
COLUMN DENSE_rank FORMAT 999
COLUMN full_name FORMAT A30
COLUMN job_title FORMAT A20
COLUMN total FORMAT A15
COLUMN percentage FORMAT A10

-- Select the top 10 employees based on total sales
SELECT
    DENSE_rank,
    full_name,
    job_title,
    total,
    percentage || '%' AS percentage
FROM
    Employees_Details
WHERE
    DENSE_rank <= 10;

CLEAR COLUMNS
CLEAR BREAK
CLEAR COMPUTES
TTITLE OFF
```

Output:



```
Top 10 Employees by Sales Performance with Dense Rank
Percentage of Total Sales: 2022 - Q3
DENSE_RANK | FULL_NAME                      | JOB_TITLE            | TOTAL           | PERCENTAGE
---------- | ------------------------------ | -------------------- | --------------- | ----------
         1 | GERARD HERNANDEZ               | Sales Rep            |      460,230.71 |   13.50%
         2 | PAMELA CASTILLO                | Sales Rep            |      456,569.38 |   13.39%
         3 | BARRY JONES                    | Sales Rep            |      327,842.19 |    9.62%
         4 | LESLIE JENNINGS                | Sales Rep            |      295,342.64 |    8.66%
         5 | LOUI BONDUR                    | Sales Rep            |      290,301.76 |    8.51%
         6 | JULIE FIRRELLI                 | Sales Rep            |      259,719.55 |    7.62%
         7 | PETER MARSH                    | Sales Rep            |      190,057.30 |    5.57%
         8 | GEORGE VANAUF                  | Sales Rep            |      190,001.31 |    5.57%
         9 | MAMI NISHI                     | Sales Rep            |      189,942.36 |    5.57%
        10 | LARRY BOTT                     | Sales Rep            |      180,839.40 |    5.30%

10 rows selected.
```

3.1.2 Festive Sales by product line

This script prompts the user for a year and festive description, creates a view that calculates sales data and cumulative distribution, and then retrieves and displays the results in a tabular format.

The cumulative_distribution is the cumulative distribution of total sales for each product line for the specified year and festive description. The cumulative distribution values will be specific to the year user input. It allows the company to see how sales are distributed across different product lines. This can highlight which product lines are major contributors to overall festive sales and which ones have a smaller impact.

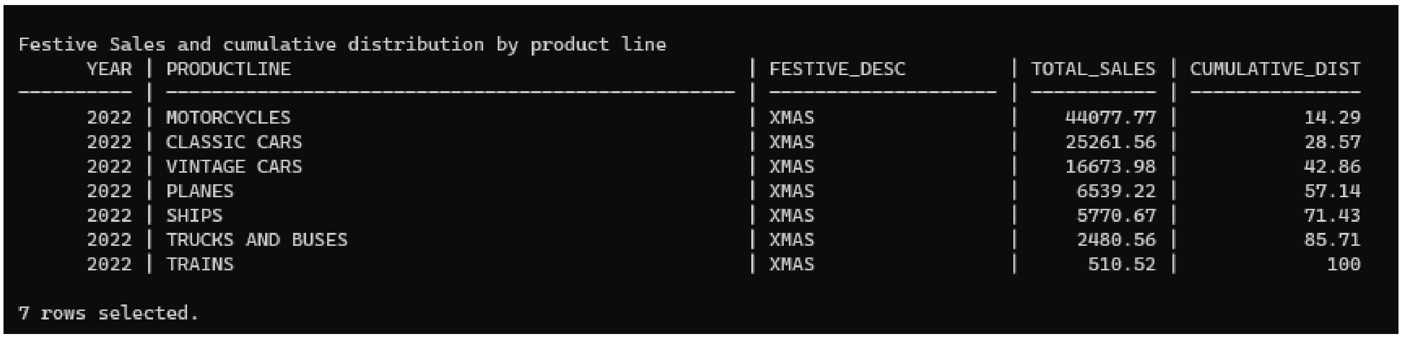
This query helps the company make informed decisions about its festive season sales strategies, product offerings, and resource allocation. The company can tailor its promotions, advertisements, and product offerings to align with the specific festive themes and customer preferences. It enables the company to better understand its sales performance during festive periods and make adjustments to improve profitability and customer satisfaction.

```
Script:
```

```
-- Accept user input for the year and festive description
ACCEPT v_cal_year CHAR PROMPT 'Enter the year (2021 - 2023): '
DEFAULT '2023'
ACCEPT v_festive_desc CHAR PROMPT 'Enter the festive
description(Eg. XMAS): ' DEFAULT 'XMAS'


TTITLE LEFT "Festive Sales and cumulative distribution by
product line";
SET SERVEROUTPUT ON
SET PAGESIZE 100
SET LINESIZE 500
SET HEADING ON
SET UNDERLINE ON
SET COLSEP ' | '

-- Create or replace a view with parameters, cumulative
distribution, and without customer state filter
CREATE OR REPLACE VIEW Sales_By_Year_Festive AS
SELECT
    dd.year,
    pd.productLine,
    dd.festive_desc,
    SUM(sf.totalAmount) AS total_sales,
    ROUND(CUME_DIST() OVER (PARTITION BY dd.year ORDER BY
SUM(sf.totalAmount) DESC) * 100, 2) AS cumulative_dist
FROM
    date_dim dd
JOIN
    sales_fact sf ON dd.date_key = sf.dateKey
JOIN
    product_dim pd ON sf.productKey = pd.productKey
WHERE
    dd.year = '&v_cal_year'
    AND dd.festive_desc = '&v_festive_desc'
GROUP BY
    dd.year, pd.productLine, dd.festive_desc;

SELECT * FROM Sales_By_Year_Festive;


CLEAR COLUMNS
CLEAR BREAK
CLEAR COMPUTES
```

```
TTITLE OFF
```

```
Output:
```



```
Festive Sales and cumulative distribution by product line
      YEAR | PRODUCTLINE                                        | FESTIVE_DESC         | TOTAL_SALES | CUMULATIVE_DIST
---------- | -------------------------------------------------- | -------------------- | ----------- | ---------------
      2022 | MOTORCYCLES                                        | XMAS                 |    44077.77 |           14.29
      2022 | CLASSIC CARS                                       | XMAS                 |    25261.56 |           28.57
      2022 | VINTAGE CARS                                       | XMAS                 |    16673.98 |           42.86
      2022 | PLANES                                             | XMAS                 |     6539.22 |           57.14
      2022 | SHIPS                                              | XMAS                 |     5770.67 |           71.43
      2022 | TRUCKS AND BUSES                                   | XMAS                 |     2480.56 |           85.71
      2022 | TRAINS                                             | XMAS                 |      510.52 |             100

7 rows selected.
```

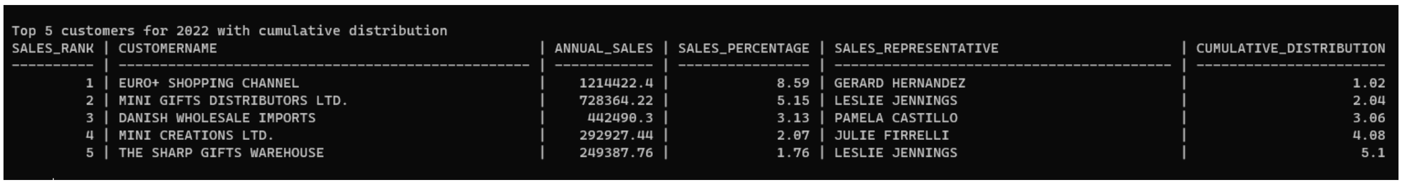
3.1.3 Top 5 customer annual sales

This script accepts user input for a specific year, calculates the top 5 customers for that year based on their total annual sales, and provides information about their sales performance, including cumulative distribution. It also associates each customer with their respective sales representative and is useful for identifying and recognizing the top-performing customers in a given year, allowing the company to focus on building and maintaining strong relationships with these key clients.
This query helps the company by providing insights into its most valuable customers for a specific year, enabling targeted marketing efforts, personalized customer service, and potential loyalty programs to nurture and retain these top clients. Additionally, it aids in evaluating the performance of sales representatives in managing these high-value customer relationships and can inform resource allocation decisions based on sales concentration among the top customers.

```
Script:
```

```
-- Accept user input for the year and festive description
ACCEPT v_cal_year CHAR PROMPT 'Enter the year (2021 - 2023): '
DEFAULT '2023'

TTITLE LEFT "Top 5 customers for &v_cal_year with cumulative
distribution";
SET SERVEROUTPUT ON
SET PAGESIZE 100
SET LINESIZE 500
SET HEADING ON
SET UNDERLINE ON
SET COLSEP ' | '

-- Create or replace a view with parameters, cumulative
distribution, and without customer state filter
CREATE OR REPLACE VIEW Top5_Cust AS
WITH RankedCustomers AS (
    SELECT
        ROW_NUMBER() OVER (PARTITION BY dd.year ORDER BY
SUM(sf.totalAmount) DESC) AS sales_rank,
        cd.customerName,
        SUM(sf.totalAmount) AS annual_sales,
        TO_NUMBER(TO_CHAR(SUM(sf.totalAmount) /
SUM(SUM(sf.totalAmount)) OVER (PARTITION BY dd.year) * 100,
'999.99')) AS sales_percentage,
        ed.firstName || ' ' || ed.lastName AS
sales_representative,
        TO_NUMBER(TO_CHAR(CUME_DIST() OVER (PARTITION BY
dd.year ORDER BY SUM(sf.totalAmount) DESC) * 100, '999.99')) AS
cumulative_distribution
    FROM
        date_dim dd
    JOIN
        sales_fact sf ON dd.date_key = sf.dateKey
    JOIN
        customer_dim cd ON sf.customerKey = cd.customerKey
    LEFT JOIN
        employee_dim ed ON cd.salesRepEmployeeNumber =
ed.employeeNumber
    WHERE
        dd.year = '&v_cal_year' -- Use the user-provided year
    GROUP BY
        dd.year, cd.customerName, ed.firstName, ed.lastName
)
SELECT *
FROM RankedCustomers
WHERE sales_rank <= 5
ORDER BY annual_sales DESC;

select * from Top5_Cust;

CLEAR COLUMNS
CLEAR BREAK
CLEAR COMPUTES
TTITLE OFF
```

Output:



```
Top 5 customers for 2022 with cumulative distribution
SALES_RANK | CUSTOMERNAME                                       | ANNUAL_SALES | SALES_PERCENTAGE | SALES_REPRESENTATIVE                          | CUMULATIVE_DISTRIBUTION
---------- | -------------------------------------------------- | ------------ | ---------------- | --------------------------------------------- | -----------------------
         1 | EURO+ SHOPPING CHANNEL                             |    1214422.4 |             8.59 | GERARD HERNANDEZ                              |                    1.02
         2 | MINI GIFTS DISTRIBUTORS LTD.                       |    728364.22 |             5.15 | LESLIE JENNINGS                               |                    2.04
         3 | DANISH WHOLESALE IMPORTS                           |     442490.3 |             3.13 | PAMELA CASTILLO                               |                    3.06
         4 | MINI CREATIONS LTD.                                |    292927.44 |             2.07 | JULIE FIRRELLI                                |                    4.08
         5 | THE SHARP GIFTS WAREHOUSE                          |    249387.76 |             1.76 | LESLIE JENNINGS                               |                     5.1
```

3.2 Tham Hiu Huen

3.2.1 Gender Segmentation Sales Analysis Report of Product

This report show the Top 5 Male and Female Higher Different of product and its corresponding productline. By knowing the product that is more preferred by gender, the company could adjust the strategic of advertisment and promotion.

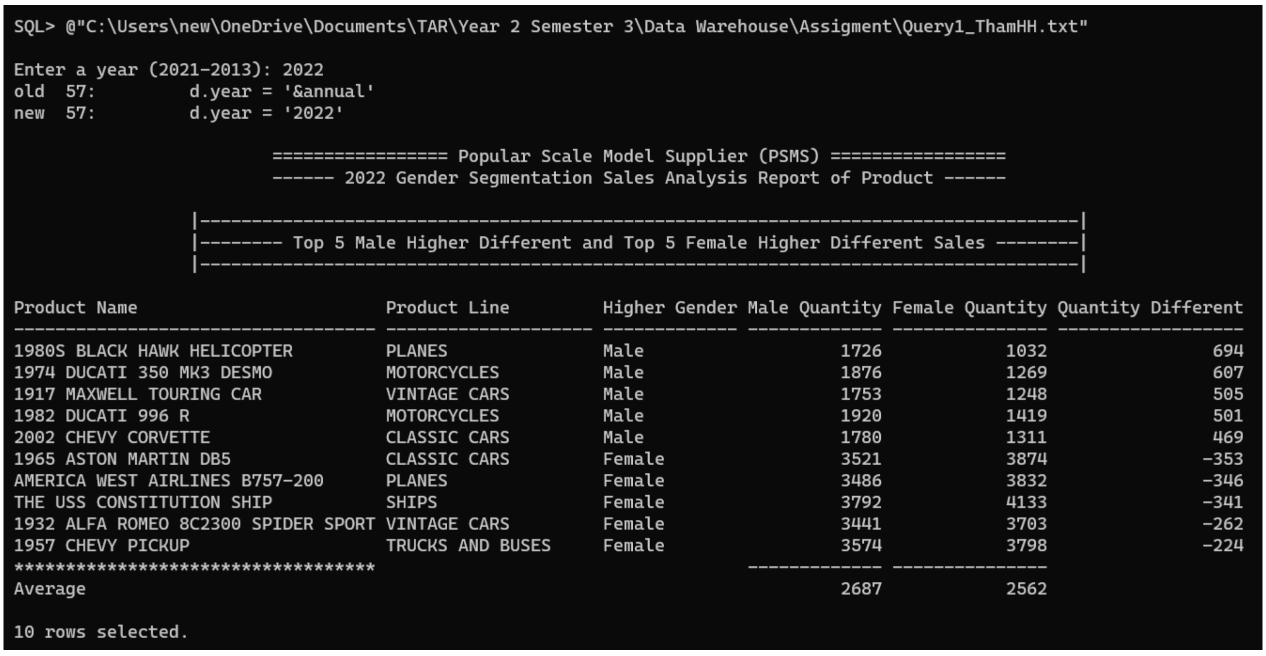
This report provides a thorough examination of how PSMS products perform in terms of sales among different genders. It carefully dissects the sales data, separating it into male and female categories, and pinpoints which products perform the best within each category. This kind of analysis arms PSMS with practical insights that can be used to fine-tune their marketing and product strategies. Additionally, the report offers a useful benchmark by providing average sales quantities for both males and females, helping PSMS gauge broader gender-based sales trends. By leveraging these insights, PSMS gains a competitive edge in the scale model industry. They can make well-informed decisions, tailor their products and marketing approaches, and ultimately strengthen their position in the market.

Code:

```
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF

SET linesize 120
SET pagesize 50

column productName heading 'Product Name' format A35
column productLine heading 'Product Line' format A20
column higherQuantityGender heading 'Higher Gender' format A13
column quantityorderedmale heading 'Male Quantity' format
99999999
column averageMaleAge heading 'AVG Male Age' format 99
column quantityorderedfemale heading 'Female Quantity' format
999999
column averageFemeleAge heading 'AVG Femele Age' format 99
column quantityDifference heading 'Quantity Different'  format
999999

Prompt
ACCEPT annual FORMAT 'A11' PROMPT 'Enter a year (2021-2013): ';

TTITLE CENTER '================= Popular Scale Model Supplier
(PSMS) =================' SKIP 1 -
CENTER '------ 'annual ' Gender Segmentation Sales Analysis
Report of Product ------' SKIP 2 -
CENTER
'|---------------------------------------------------------------
-----------------------|' SKIP 1 -
CENTER '|-------- Top 5 Male Higher Different and Top 5 Female
Higher Different Sales --------|' SKIP 1 -
CENTER
```

```
'|------------------------------------------------------------
------------------------|' SKIP 2

BREAK ON productName ON REPORT
COMPUTE AVG LABEL 'Average' OF quantityorderedmale ON REPORT
COMPUTE AVG LABEL 'Average' OF quantityorderedfemale ON REPORT

SELECT *
FROM (
    SELECT
        p.productName, p.productLine,
        CASE
            WHEN SUM(CASE WHEN c.gender = 'M' THEN
sf.quantityOrdered ELSE 0 END) >
                 SUM(CASE WHEN c.gender = 'F' THEN
sf.quantityOrdered ELSE 0 END) THEN 'Male'
            WHEN SUM(CASE WHEN c.gender = 'M' THEN
sf.quantityOrdered ELSE 0 END) <
                 SUM(CASE WHEN c.gender = 'F' THEN
sf.quantityOrdered ELSE 0 END) THEN 'Female'
            ELSE 'Equal'
        END AS higherQuantityGender,
        SUM(CASE WHEN c.gender = 'M' THEN sf.quantityOrdered
ELSE 0 END) AS quantityOrderedMale,
        SUM(CASE WHEN c.gender = 'F' THEN sf.quantityOrdered
ELSE 0 END) AS quantityOrderedFemale,
        SUM(CASE WHEN c.gender = 'M' THEN sf.quantityOrdered
ELSE 0 END) -
        SUM(CASE WHEN c.gender = 'F' THEN sf.quantityOrdered
ELSE 0 END) AS quantityDifference
    FROM
        product_dim p
    JOIN
        sales_fact sf ON p.productKey = sf.productKey
    JOIN
        customer_dim c ON sf.customerKey = c.customerKey
    JOIN
        date_dim d ON sf.dateKey = d.date_Key
    WHERE
        d.year = 2023
    GROUP BY
        p.productName, p.productLine
    ORDER BY
        quantityDifference DESC
) result_table
WHERE ROWNUM <= 5
UNION ALL
SELECT *
FROM (
    SELECT
        p.productName, p.productLine, -- Added a comma here
        CASE
            WHEN SUM(CASE WHEN c.gender = 'M' THEN
sf.quantityOrdered ELSE 0 END) >
                 SUM(CASE WHEN c.gender = 'F' THEN
sf.quantityOrdered ELSE 0 END) THEN 'Male'
            WHEN SUM(CASE WHEN c.gender = 'M' THEN
sf.quantityOrdered ELSE 0 END) <
                 SUM(CASE WHEN c.gender = 'F' THEN
sf.quantityOrdered ELSE 0 END) THEN 'Female'
            ELSE 'Equal'
        END AS higherQuantityGender,
        SUM(CASE WHEN c.gender = 'M' THEN sf.quantityOrdered
ELSE 0 END) AS quantityOrderedMale,
        SUM(CASE WHEN c.gender = 'F' THEN sf.quantityOrdered
ELSE 0 END) AS quantityOrderedFemale,
        SUM(CASE WHEN c.gender = 'M' THEN sf.quantityOrdered
ELSE 0 END) -
        SUM(CASE WHEN c.gender = 'F' THEN sf.quantityOrdered
ELSE 0 END) AS quantityDifference
    FROM
        product_dim p
    JOIN
        sales_fact sf ON p.productKey = sf.productKey
    JOIN
        customer_dim c ON sf.customerKey = c.customerKey
    JOIN
        date_dim d ON sf.dateKey = d.date_Key
    WHERE
        d.year = '&annual'
    GROUP BY
        p.productName, p.productLine
    ORDER BY
        quantityDifference
) result_table
WHERE ROWNUM <= 5;
```

Sample Output:



```
SQL> @"C:\Users\new\OneDrive\Documents\TAR\Year 2 Semester 3\Data Warehouse\Assigment\Query1_ThamHH.txt"

Enter a year (2021-2013): 2022
old  57:         d.year = '&annual'
new  57:         d.year = '2022'

                         ================= Popular Scale Model Supplier (PSMS) =================
                         ------ 2022 Gender Segmentation Sales Analysis Report of Product ------

                 |-------------------------------------------------------------------------------------|
                 |-------- Top 5 Male Higher Different and Top 5 Female Higher Different Sales --------|
                 |-------------------------------------------------------------------------------------|

Product Name                        Product Line         Higher Gender Male Quantity Female Quantity Quantity Different
----------------------------------- -------------------- ------------- ------------- --------------- ------------------
1980S BLACK HAWK HELICOPTER         PLANES               Male                   1726            1032                694
1974 DUCATI 350 MK3 DESMO           MOTORCYCLES          Male                   1876            1269                607
1917 MAXWELL TOURING CAR            VINTAGE CARS         Male                   1753            1248                505
1982 DUCATI 996 R                   MOTORCYCLES          Male                   1920            1419                501
2002 CHEVY CORVETTE                 CLASSIC CARS         Male                   1780            1311                469
1965 ASTON MARTIN DB5               CLASSIC CARS         Female                 3521            3874               -353
AMERICA WEST AIRLINES B757-200      PLANES               Female                 3486            3832               -346
THE USS CONSTITUTION SHIP           SHIPS                Female                 3792            4133               -341
1932 ALFA ROMEO 8C2300 SPIDER SPORT VINTAGE CARS         Female                 3441            3703               -262
1957 CHEVY PICKUP                   TRUCKS AND BUSES     Female                 3574            3798               -224
***********************************                                    ------------- ---------------
Average                                                                         2687            2562

10 rows selected.
```

3.2.2 Monthly Sales Analysis Report of Office with Holiday Count

This report number of customer, quantity sold and business volume of each month. Then, it show the different of business volumn compare to the previous month. It then show the number of holiday in each month, to analyze whether the number of holiday will affect the business column in each month.

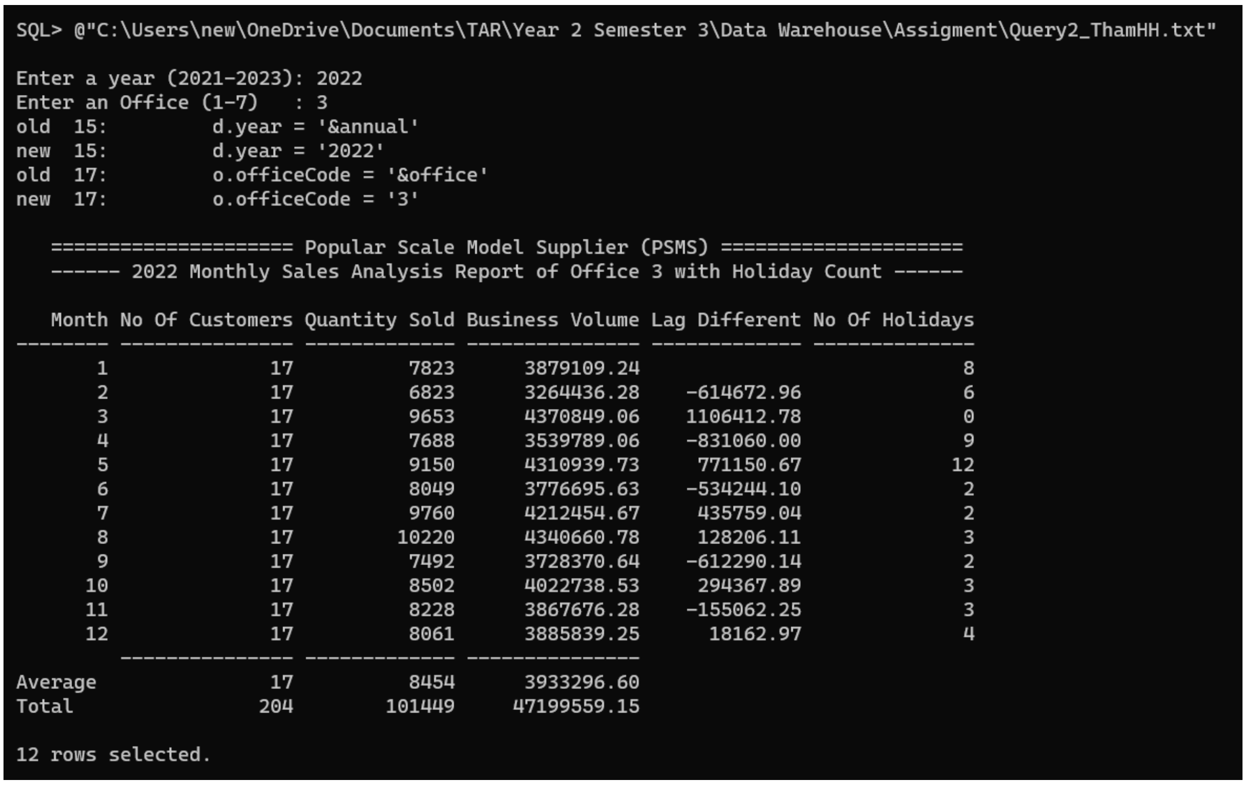
This report provides a valuable resource for PSMS management to enhance their competitiveness. By meticulously analyzing monthly sales data for a specific year and office location, it offers critical insights into customer engagement and revenue generation. The report not only tracks essential metrics such as total customers, quantity ordered, and total sales amount but also includes a comparison of monthly revenue trends. Moreover, it identifies the presence of holidays within each month, enabling PSMS to tailor their marketing strategies and inventory management to capitalize on peak seasonal demand. By leveraging this report, PSMS can make data-driven decisions, optimize resource allocation, and ultimately strengthen their competitive position within the scale model industry.

Code:

```
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF

SET linesize 85
SET pagesize 50

column month_no heading 'Month' format 9999999
column TotalCustomers heading 'No Of Customers' format 9999
column TotalQuantityOrdered heading 'Quantity Sold' format
9999999
column TotalAmount heading 'Business Volume' format 99999999.99
column LagTotalAmount heading 'Lag Different' format
99999999.99
column HolidayCount heading 'No Of Holidays' format 99

Prompt
ACCEPT annual FORMAT 'A11' PROMPT 'Enter a year (2021-2023): ';
ACCEPT office FORMAT 'A11' PROMPT 'Enter an Office (1-7)   : ';

TTITLE CENTER '==================== Popular Scale Model
Supplier (PSMS) ====================' SKIP 1 -
CENTER '------ 'annual ' Monthly Sales Analysis Report of
Office 'office' with Holiday Count ------' SKIP 2

BREAK ON month_num ON REPORT
COMPUTE SUM LABEL 'Total' AVG LABEL 'Average' OF TotalCustomers
ON REPORT
COMPUTE SUM LABEL 'Total' AVG LABEL 'Average' OF
TotalQuantityOrdered ON REPORT
COMPUTE SUM LABEL 'Total' AVG LABEL 'Average' OF TotalAmount ON
REPORT
```

```
WITH MonthlyTotals AS (
    SELECT
        d.month_no,
        COUNT(DISTINCT s.customerKey) AS TotalCustomers,
        SUM(s.quantityOrdered) AS TotalQuantityOrdered,
        SUM(s.totalAmount) AS TotalAmount,
        (SUM(s.totalAmount) - LAG(SUM(s.totalAmount)) OVER
(ORDER BY d.month_no)) AS LagTotalAmount
    FROM
        date_dim d
    LEFT JOIN
        sales_fact s ON d.date_key = s.dateKey
    JOIN
        office_dim o ON s.officeKey = o.officeKey
    WHERE
        d.year = '&annual'
    AND
        o.officeCode = '&office'
    GROUP BY
        d.month_no
    ORDER BY
        d.month_no
)
SELECT
    M.*,
    (SELECT COUNT(*) FROM date_dim d WHERE d.month_no =
M.month_no AND d.holiday_ind = 'Y') AS HolidayCount
FROM
    MonthlyTotals M
ORDER BY
    M.month_no;
```

Sample Output:



```
SQL> @"C:\Users\new\OneDrive\Documents\TAR\Year 2 Semester 3\Data Warehouse\Assigment\Query2_ThamHH.txt"

Enter a year (2021-2023): 2022
Enter an Office (1-7)   : 3
old  15:        d.year = '&annual'
new  15:        d.year = '2022'
old  17:        o.officeCode = '&office'
new  17:        o.officeCode = '3'

   ==================== Popular Scale Model Supplier (PSMS) ====================
   ------ 2022 Monthly Sales Analysis Report of Office 3 with Holiday Count ------

   Month No Of Customers Quantity Sold Business Volume Lag Different No Of Holidays
-------- --------------- ------------- --------------- ------------- --------------
       1              17          7823      3879109.24                            8
       2              17          6823      3264436.28    -614672.96              6
       3              17          9653      4370849.06    1106412.78              0
       4              17          7688      3539789.06    -831060.00              9
       5              17          9150      4310939.73     771150.67             12
       6              17          8049      3776695.63    -534244.10              2
       7              17          9760      4212454.67     435759.04              2
       8              17         10220      4340660.78     128206.11              3
       9              17          7492      3728370.64    -612290.14              2
      10              17          8502      4022738.53     294367.89              3
      11              17          8228      3867676.28    -155062.25              3
      12              17          8061      3885839.25      18162.97              4
         --------------- ------------- ---------------
Average               17          8454      3933296.60
Total                204        101449     47199559.15

12 rows selected.
```

3.2.3 Quarter Sales Analysis Report of Product Line

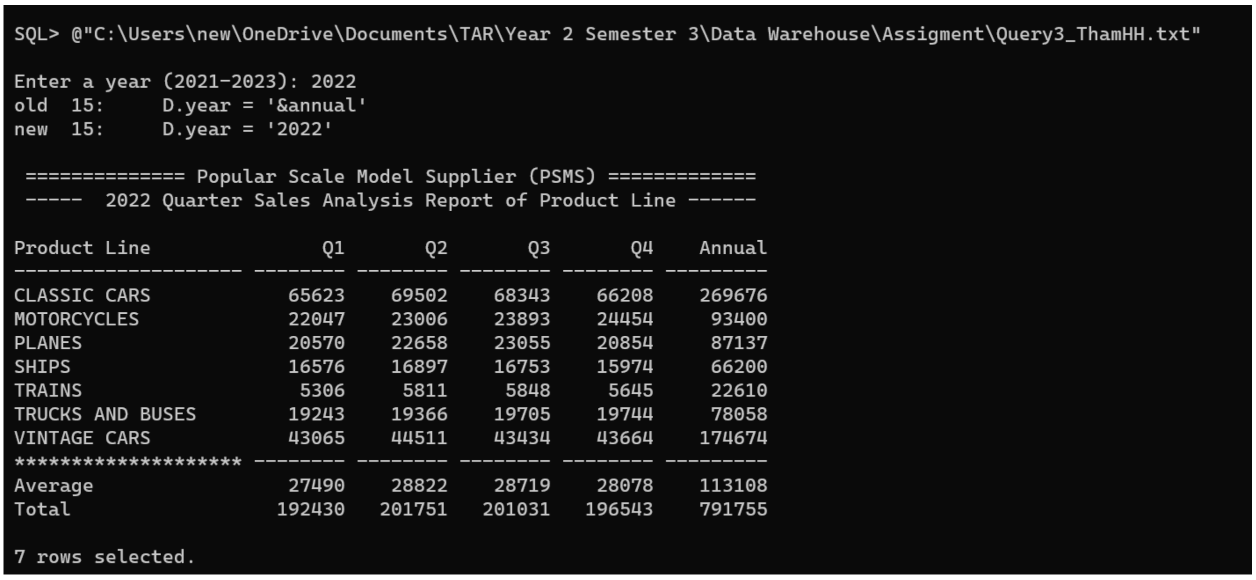
This report provide show the business volumn of each quarter and the whole annual business volume of each productline.

This report plays a pivotal role in enhancing PSMS's competitive edge. It meticulously examines quarterly sales data for various product lines across the years 2021 to 2023. By breaking down sales figures into individual quarters (Q1, Q2, Q3, Q4) and providing a total quantity summary, the report equips PSMS management with crucial insights into seasonal sales trends. Armed with this information, PSMS can strategically adjust their production schedules, marketing campaigns, and inventory management to align with the ebb and flow of customer demand. This adaptability not only optimizes resource allocation but also allows PSMS to stay agile in response to market dynamics, ultimately strengthening their position and competitiveness within the scale model industry.

Code:

```
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES
TTITLE OFF

SET linesize 66
SET pagesize 50

-- Format the AVG columns to display two decimal places
COLUMN Q1 HEADING 'Q1' FORMAT 9999999
COLUMN Q2 HEADING 'Q2' FORMAT 9999999
COLUMN Q3 HEADING 'Q3' FORMAT 9999999
COLUMN Q4 HEADING 'Q4' FORMAT 9999999
COLUMN TOTALQUANTITY HEADING 'Total Quantity' FORMAT 99999999
column PRODUCTLINE heading 'Product Line' format A20
column TOTALQUANTITY heading 'Annual' format 99999999

Prompt
ACCEPT annual FORMAT 'A11' PROMPT 'Enter a year (2021-2023): ';

TTITLE CENTER '============= Popular Scale Model Supplier
(PSMS) =============' SKIP 1 -
CENTER '-----  'annual ' Quarter Sales Analysis Report of
Product Line ------' SKIP 2

BREAK ON productLine ON REPORT
COMPUTE SUM LABEL 'Total' AVG LABEL 'Average' OF Q1 ON REPORT
COMPUTE SUM LABEL 'Total' AVG LABEL 'Average' OF Q2 ON REPORT
COMPUTE SUM LABEL 'Total' AVG LABEL 'Average' OF Q3 ON REPORT
COMPUTE SUM LABEL 'Total' AVG LABEL 'Average' OF Q4 ON REPORT
COMPUTE SUM LABEL 'Total' AVG LABEL 'Average' OF TOTALQUANTITY
ON REPORT

SELECT
    P.productLine,
    SUM(CASE WHEN D.quarter = 'Q1' THEN S.quantityOrdered ELSE 0
END) AS "Q1",
    SUM(CASE WHEN D.quarter = 'Q2' THEN S.quantityOrdered ELSE 0
END) AS "Q2",
    SUM(CASE WHEN D.quarter = 'Q3' THEN S.quantityOrdered ELSE 0
END) AS "Q3",
    SUM(CASE WHEN D.quarter = 'Q4' THEN S.quantityOrdered ELSE 0
END) AS "Q4",
    SUM(S.quantityOrdered) AS TotalQuantity
FROM
    sales_fact S
JOIN
    date_dim D ON S.dateKey = D.date_key
JOIN
    product_dim P ON S.productKey = P.productKey
WHERE
    D.year = '&annual'
GROUP BY
    P.productLine
ORDER BY
    P.productLine;
```

Sample Output:



```
SQL> @"C:\Users\new\OneDrive\Documents\TAR\Year 2 Semester 3\Data Warehouse\Assigment\Query3_ThamHH.txt"

Enter a year (2021-2023): 2022
old  15:     D.year = '&annual'
new  15:     D.year = '2022'

 ============= Popular Scale Model Supplier (PSMS) ============
 -----  2022 Quarter Sales Analysis Report of Product Line ------

Product Line               Q1       Q2       Q3       Q4    Annual
-------------------- -------- -------- -------- -------- ---------
CLASSIC CARS            65623    69502    68343    66208    269676
MOTORCYCLES             22047    23006    23893    24454     93400
PLANES                  20570    22658    23055    20854     87137
SHIPS                   16576    16897    16753    15974     66200
TRAINS                   5306     5811     5848     5645     22610
TRUCKS AND BUSES        19243    19366    19705    19744     78058
VINTAGE CARS            43065    44511    43434    43664    174674
******************** -------- -------- -------- -------- ---------
Average                 27490    28822    28719    28078    113108
Total                  192430   201751   201031   196543    791755

7 rows selected.
```

3.3 Koong Jie Lum

3.3.1 Sales Report of Selected Product Line in Selected Year

Purpose: This query generates a detailed sales report for the specified product line in the given year. It provides insights into monthly sales performance, allowing stakeholders to identify patterns, trends, and areas for improvement. The report includes essential metrics such as total sales amount, growth rate, cumulative distribution, and percentile rank.

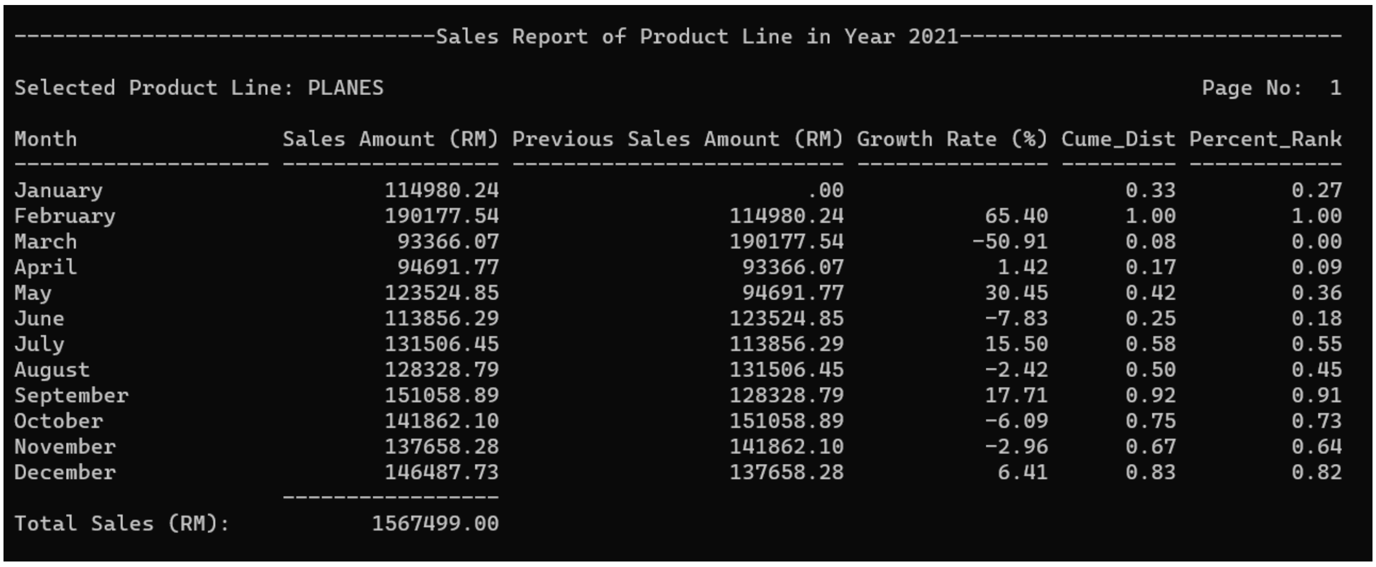
The LAG() function is used to access the previous month's total sales amount, allowing a month-to-month comparison. CUME_DIST() computes the cumulative distribution of sales amounts, assigning a relative rank to each month's sales. It assists in understanding the position of each month's sales within the entire dataset, showcasing the proportion of sales values below the current month. This aids in identifying exceptional sales months in the context of the entire year. PERCENT_RANK() assigns a percentile rank to each month's sales amount within the ordered set. It provides a normalized ranking, considering potential tied values. This helps in determining the relative competitiveness of each month's sales performance compared to others.

Script:

```
/* Product Line:
   TRUCKS AND BUSES
   PLANES
   CLASSIC CARS
   MOTORCYCLES
   TRAINS
   VINTAGE CARS
   SHIPS
*/

SET linesize 104
SET pagesize 100

COLUMN Month FORMAT A20 HEADING "Month";
COLUMN Sales_Amount FORMAT 99999999.99 HEADING "Sales Amount (RM)";
COLUMN Prev_Sales_Amount FORMAT 99999999.99 HEADING "Previous Sales
Amount (RM)";
COLUMN Growth_Rate FORMAT 9999.99 HEADING "Growth Rate (%)";
COLUMN Cume_Dist FORMAT 0.99 HEADING "Cume_Dist";
COLUMN Percent_Rank FORMAT 0.99 HEADING "Percent_Rank";

ACCEPT v_year NUMBER FORMAT 9999 PROMPT 'Enter a Year: '
ACCEPT v_productLine CHAR   FORMAT 'A50' PROMPT 'Enter a Product Line:
'

TTITLE CENTER '--------------------------------Sales Report of
Product Line in Year
'&v_year'------------------------------------------------------------
-' SKIP 2 LEFT 'Selected Product Line: '&v_productLine'' RIGHT 'Page
No: ' FORMAT 9 SQL.PNO SKIP 2

BREAK ON REPORT SKIP 2;
COMPUTE SUM LABEL 'Total Sales (RM): ' OF Sales_Amount ON REPORT
```

```
WITH cte_MonthSales AS (
   SELECT
      TO_DATE(d.month, 'MM') AS sales_month,
      SUM(sf.totalAmount) AS sales_amount,
      LAG(SUM(sf.totalAmount), 1, 0) OVER (ORDER BY TO_DATE(d.month,
   'MM')) AS prev_sales_amount
   FROM sales_fact sf
   JOIN date_dim d ON sf.dateKey = d.date_key
   JOIN product_dim p ON sf.productKey = p.productKey
   WHERE d.year = '&v_year'
   AND p.productLine = UPPER('&v_productLine')
   GROUP BY TO_DATE(d.month, 'MM')
)
SELECT
   TO_CHAR(sales_month, 'Month') AS Month,
   sales_amount AS Sales_Amount,
   prev_sales_amount AS Prev_Sales_Amount,
   CASE
      WHEN EXTRACT(MONTH FROM sales_month) = 1 THEN NULL
      ELSE ROUND(((sales_amount - prev_sales_amount) /
   NULLIF(prev_sales_amount, 0)) * 100, 2)
   END AS Growth_Rate,
   CUME_DIST() OVER (ORDER BY sales_amount) AS Cume_Dist,
   PERCENT_RANK() OVER (ORDER BY sales_amount) AS Percent_Rank
FROM cte_MonthSales
ORDER BY sales_month;

CLEAR COMPUTE
CLEAR COLUMNS
CLEAR BREAKS
TTITLE OFF
```

Output:



```
--------------------------------Sales Report of Product Line in Year 2021------------------------------

Selected Product Line: PLANES                                                              Page No:  1

Month                Sales Amount (RM) Previous Sales Amount (RM) Growth Rate (%) Cume_Dist Percent_Rank
-------------------- ----------------- -------------------------- --------------- --------- ------------
January                      114980.24                        .00                      0.33         0.27
February                     190177.54                  114980.24           65.40      1.00         1.00
March                         93366.07                  190177.54          -50.91      0.08         0.00
April                         94691.77                   93366.07            1.42      0.17         0.09
May                          123524.85                   94691.77           30.45      0.42         0.36
June                         113856.29                  123524.85           -7.83      0.25         0.18
July                         131506.45                  113856.29           15.50      0.58         0.55
August                       128328.79                  131506.45           -2.42      0.50         0.45
September                    151058.89                  128328.79           17.71      0.92         0.91
October                      141862.10                  151058.89           -6.09      0.75         0.73
November                     137658.28                  141862.10           -2.96      0.67         0.64
December                     146487.73                  137658.28            6.41      0.83         0.82
                     -----------------
Total Sales (RM):           1567499.00
```

3.3.2 Report of Profitable Products in Selected Year and Quarter

Purpose: This report is to provide insights into the top 5 most profitable products for the selected year and quarter, helping businesses identify their best-performing items during this period. Through clear presentation of product codes, names, monthly profits, and total quarterly profits, stakeholders gain rapid access to critical performance data.

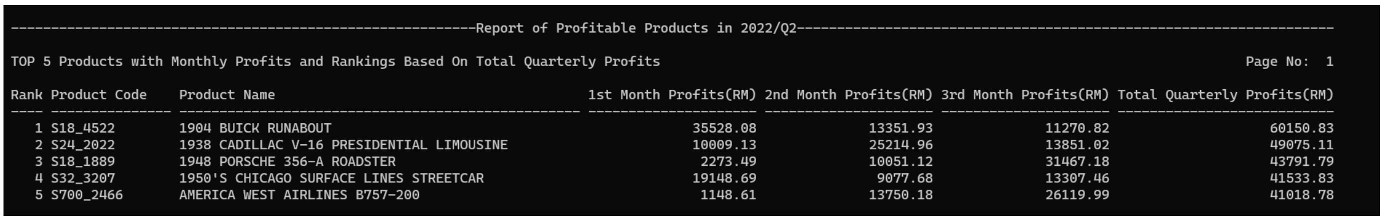
The report leverages PIVOT() to break down profits for each month within the selected quarter. Utilizing DENSE_RANK(), it accurately ranks products based on their total quarterly profits.

```
Script:
```

```
SET LINESIZE 165
SET PAGESIZE 100

ACCEPT v_year NUMBER FORMAT '9999' PROMPT 'Enter a year: '
ACCEPT v_quarter CHAR FORMAT 'A2' PROMPT 'Enter a quarter (Q1, Q2, Q3,
Q4): '

COLUMN ProfitRank    FORMAT 999        HEADING 'Rank';
COLUMN productCode   FORMAT A15        HEADING 'Product Code';
COLUMN productName   FORMAT A50        HEADING 'Product Name';
COLUMN Month1        FORMAT 999999.99  HEADING '1st Month
Profits(RM)';
COLUMN Month2        FORMAT 999999.99  HEADING '2nd Month
Profits(RM)';
COLUMN Month3        FORMAT 999999.99  HEADING '3rd Month
Profits(RM)';
COLUMN QuarterProfit FORMAT 9999999.99 HEADING 'Total Quarterly
Profits(RM)';

TTITLE CENTER
'-----------------------------------------------------------Report of
Profitable Products in
'&v_year'/'v_quarter'-----------------------------------------------
------------------------' SKIP 2 LEFT 'TOP 5 Products with Monthly
Profits and Rankings Based On Total Quarterly Profits' RIGHT 'Page No:
' FORMAT 9 SQL.PNO SKIP 2

WITH MonthProfit AS (
   SELECT p.productCode,
      p.productName,
      d.month,
      SUM(sf.totalAmount - (sf.quantityOrdered * p.buyPrice)) AS
   profit
   FROM sales_fact sf
   JOIN product_dim p ON sf.productKey = p.productKey
   JOIN date_dim d ON sf.dateKey = d.date_key
   WHERE d.year = '&v_year'
   GROUP BY p.productCode, p.productName, d.month
```

```
),
MonthlyPivot AS (
   SELECT productCode,
      productName,
      CASE
         WHEN '&v_quarter' = 'Q1' THEN January
         WHEN '&v_quarter' = 'Q2' THEN April
         WHEN '&v_quarter' = 'Q3' THEN July
         WHEN '&v_quarter' = 'Q4' THEN October
      END AS Month1,
      CASE
         WHEN '&v_quarter' = 'Q1' THEN February
         WHEN '&v_quarter' = 'Q2' THEN May
         WHEN '&v_quarter' = 'Q3' THEN August
         WHEN '&v_quarter' = 'Q4' THEN November
      END AS Month2,
      CASE
         WHEN '&v_quarter' = 'Q1' THEN March
         WHEN '&v_quarter' = 'Q2' THEN June
         WHEN '&v_quarter' = 'Q3' THEN September
         WHEN '&v_quarter' = 'Q4' THEN December
      END AS Month3,
      CASE
         WHEN '&v_quarter' = 'Q1' THEN COALESCE(January, 0) +
   COALESCE(February, 0) + COALESCE(March, 0)
         WHEN '&v_quarter' = 'Q2' THEN COALESCE(April, 0) +
   COALESCE(May, 0) + COALESCE(June, 0)
         WHEN '&v_quarter' = 'Q3' THEN COALESCE(July, 0) +
   COALESCE(August, 0) + COALESCE(September, 0)
         WHEN '&v_quarter' = 'Q4' THEN COALESCE(October, 0) +
   COALESCE(November, 0) + COALESCE(December, 0)
      END AS QuarterProfit
   FROM MonthProfit
   PIVOT (
      SUM(profit) FOR month IN (
         'January' AS January,
         'February' AS February,
         'March' AS March,
         'April' AS April,
         'May' AS May,
         'June' AS June,
         'July' AS July,
         'August' AS August,
         'September' AS September,
         'October' AS October,
         'November' AS November,
         'December' AS December
      )
   )
)
SELECT *
FROM (
   SELECT DENSE_RANK() OVER (ORDER BY QuarterProfit DESC) AS
   ProfitRank,
      productCode,
      productName,
      Month1,
      Month2,
      Month3,
      QuarterProfit
   FROM MonthlyPivot
)
WHERE ProfitRank <= 5;

CLEAR COLUMNS
TTITLE OFF
```

```
   Output:
```



```
----------------------------------------------------------Report of Profitable Products in 2022/Q2-------------------------------------------------------------------

TOP 5 Products with Monthly Profits and Rankings Based On Total Quarterly Profits                                                                         Page No:  1

Rank Product Code    Product Name                                       1st Month Profits(RM) 2nd Month Profits(RM) 3rd Month Profits(RM) Total Quarterly Profits(RM)
---- --------------- -------------------------------------------------- --------------------- --------------------- --------------------- ---------------------------
   1 S18_4522        1904 BUICK RUNABOUT                                             35528.08              13351.93              11270.82                    60150.83
   2 S24_2022        1938 CADILLAC V-16 PRESIDENTIAL LIMOUSINE                       10009.13              25214.96              13851.02                    49075.11
   3 S18_1889        1948 PORSCHE 356-A ROADSTER                                      2273.49              10051.12              31467.18                    43791.79
   4 S32_3207        1950'S CHICAGO SURFACE LINES STREETCAR                          19148.69               9077.68              13307.46                    41533.83
   5 S700_2466       AMERICA WEST AIRLINES B757-200                                   1148.61              13750.18              26119.99                    41018.78
```

3.3.3 Customer Purchase Patterns Analysis

Purpose: This query analyzes customer purchase patterns, focusing on the top 5 customers with the shortest average intervals between purchases. It presents essential details including customer key, name, first and last purchase dates, total number of purchases, and the average number of days between purchases. The analysis is aimed at identifying the most active and consistent buyers, providing valuable insights for targeted marketing strategies and customer relationship management.

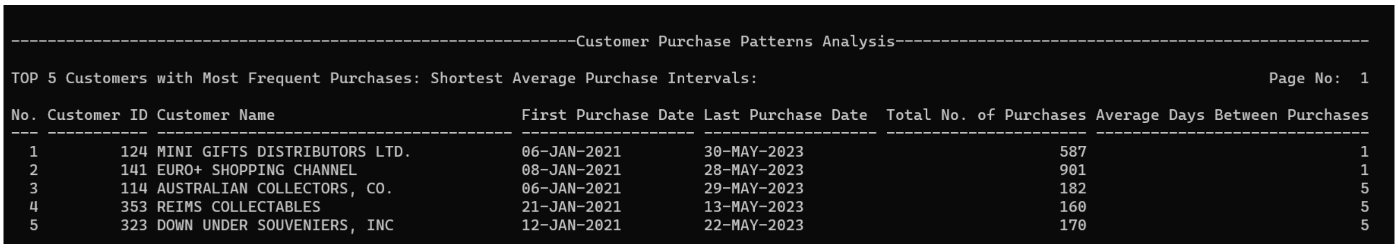
The LAG() function is utilized to access the previous purchase date, enabling the calculation of time intervals between consecutive purchases, while the LEAD() function is employed to access the next purchase date. By leveraging these functions, the query efficiently calculates the average intervals between purchases. This query also employs ROW_NUMBER() to track the order of purchases made by each customer. By subtracting 1 from the maximum purchase_sequence, the query calculates the total number of purchases for each customer.

```
Script:
```

```
SET linesize 149
SET pagesize 100
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';

COLUMN no                          FORMAT 9      HEADING 'No.';
COLUMN customerNumber              FORMAT 999999 HEADING 'Customer ID';
COLUMN customerName                FORMAT A39    HEADING 'Customer
Name';
COLUMN first_purchase_date         FORMAT A19    HEADING 'First
Purchase Date';
COLUMN last_purchase_date          FORMAT A19    HEADING 'Last Purchase
Date';
COLUMN total_purchases             FORMAT 9999   HEADING 'Total No. of
Purchases';
COLUMN avg_days_between_purchases FORMAT 999    HEADING 'Average Days
Between Purchases';

TTITLE CENTER
```

```
'-------------------------------------------------------------Custome
r Purchase Patterns
Analysis---------------------------------------------------------------
----------------' SKIP 2 LEFT 'TOP 5 Customers with Most Frequent
Purchases: Shortest Average Purchase Intervals: ' RIGHT 'Page No: '
FORMAT 9 SQL.PNO SKIP 2

WITH cte_PurchasePatterns AS (
   SELECT customerKey,
      dateKey,
      LAG(dateKey) OVER (PARTITION BY customerKey ORDER BY dateKey) AS
   prev_purchase_date,
      LEAD(dateKey) OVER (PARTITION BY customerKey ORDER BY dateKey)
   AS next_purchase_date,
      ROW_NUMBER() OVER (PARTITION BY customerKey ORDER BY dateKey) AS
   purchase_sequence,
      CASE
         WHEN LAG(dateKey) OVER (PARTITION BY customerKey ORDER BY
   dateKey) IS NOT NULL
                 AND LEAD(dateKey) OVER (PARTITION BY customerKey
   ORDER BY dateKey) IS NOT NULL
            THEN 1
            ELSE 0
         END AS alternating_purchase_flag
   FROM sales_fact
)
SELECT ROWNUM AS no, pp.*
FROM (
   SELECT
      c.customerNumber,
      c.customerName,
      MIN(d.cal_date) AS first_purchase_date,
      MAX(d.cal_date) AS last_purchase_date,
      MAX(pp.purchase_sequence)-1 AS total_purchases,
      ROUND(AVG(pp.dateKey - pp.prev_purchase_date), 0) AS
   avg_days_between_purchases
    FROM cte_PurchasePatterns pp
    JOIN date_dim d ON pp.dateKey = d.date_key
    JOIN customer_dim c ON pp.customerKey = c.customerKey
    WHERE pp.alternating_purchase_flag = 1
    GROUP BY c.customerNumber, c.customerName
    HAVING COUNT(*) >= 2
    ORDER BY avg_days_between_purchases
) pp
WHERE ROWNUM <= 5;

CLEAR COLUMNS
TTITLE OFF
```

Output:



```
-------------------------------------------------------------Customer Purchase Patterns Analysis------------------------------------------------------

TOP 5 Customers with Most Frequent Purchases: Shortest Average Purchase Intervals:                                                        Page No:   1

No. Customer ID Customer Name                            First Purchase Date Last Purchase Date  Total No. of Purchases Average Days Between Purchases
--- ----------- ---------------------------------------- ------------------- ------------------- ---------------------- ------------------------------
  1         124 MINI GIFTS DISTRIBUTORS LTD.             06-JAN-2021         30-MAY-2023                            587                              1
  2         141 EURO+ SHOPPING CHANNEL                   08-JAN-2021         28-MAY-2023                            901                              1
  3         114 AUSTRALIAN COLLECTORS, CO.               06-JAN-2021         29-MAY-2023                            182                              5
  4         353 REIMS COLLECTABLES                       21-JAN-2021         13-MAY-2023                            160                              5
  5         323 DOWN UNDER SOUVENIERS, INC               12-JAN-2021         22-MAY-2023                            170                              5
```

3.4 Tan Mei Yin

3.4.1 Top 5 highest profit state in 2022

This report is to analyze the top 5 states in terms of sales revenue and profit for a specific year which is in the year 2022. Profit signifies the remaining income once all expenses, debts, operational costs, and additional revenue sources. The objective of this analysis is to pinpoint the states responsible for the company's highest profit generation. In this report, it will present the top 5 states that yielded the highest profit in the year 2022, along with their cumulative costs, total sales, total quantity sold, and profit figures. The columns will be arranged in the following sequence: Rank, State, Cost, Sales, Quantity, and Profit.

Query:

```
set pagesize 200
set linesize 100

ACCEPT year NUMBER FORMAT '9999' PROMPT 'ENTER THE YEAR (YYYY): '
COLUMN RANK FORMAT '99' HEADING "RANK";
COLUMN state FORMAT A15 HEADING "State";
COLUMN Sales FORMAT '999,999,999,999.00' HEADING "TotalSales";
COLUMN cost FORMAT '999,999,999,999.00' HEADING "Cost(RM)";
COLUMN qty FORMAT '999,999,999' HEADING "Quantity";
COLUMN Profit FORMAT '999,999,999,999,999.00'HEADING "Profit";

TTITLE CENTER 'Top 5 Highest Profit State in ' &year SKIP 2

CREATE OR REPLACE VIEW topProfit AS
WITH total AS
(
      SELECT c.state,
             SUM(SF.totalAmount) AS totalSales,
             SUM(P.buyPrice * SF.quantityOrdered) AS Cost,
             SUM(SF.quantityOrdered) AS Quantity,
             SUM(SF.totalAmount - (P.buyPrice * SF.quantityOrdered)) AS profit
      FROM sales_fact SF
          JOIN date_dim D ON SF.dateKey = D.date_key
          JOIN customer_dim c ON SF.customerKey = c.customerKey
          JOIN product_dim P ON SF.productKey = P.productKey
      WHERE D.year = 2022
      GROUP BY c.state
),
ranking AS
(
    SELECT state,
            RANK() OVER (ORDER BY profit DESC) AS RANK,
            totalSales AS Sales,
            Cost AS Cost,
            Quantity AS Qty,
            profit AS Profit
    FROM total
)
SELECT RANK, state, Cost, Sales, Qty, Profit
FROM ranking
WHERE RANK <= 5;

SELECT * FROM topProfit;

spool off
```

Output:

```
                Top 5 Highest Profit State in 2022

RANK State                      Cost(RM)         TotalSales     Quantity
Profit
---- --------------- ------------------- ------------------- ------------
-----------------------
   1 Kelantan         499,888,549,043.28   51,041,494,022.00  316,418,716
-448,847,055,021.28
   2 Perlis           502,403,513,730.38   47,210,042,973.00  297,147,319
-455,193,470,757.38
   3 Melaka           537,510,239,951.86   48,645,427,595.00  311,725,199
-488,864,812,356.86
   4 Kedah            857,200,018,769.23   91,313,076,250.00  554,712,627
-765,886,942,519.23
   5 Negeri Sembilan  938,000,399,027.27   99,048,799,357.00  613,685,841
-838,951,599,670.27
```

3.4.2 Annual Sales Comparison Report

This report is to examine the annual sales comparison report between 2019 to 2023. This report serves as a tool for dissecting sales patterns throughout these years, enabling the identification of periods marked by substantial sales growth or decline. This report's structure will be organizing the columns in the order of Year, Sales, Difference(Amount).

Query:

```
SET   linesize 100

SET   pagesize 35

COLUMN year FORMAT 9999 HEADING "Year";
COLUMN totalSales FORMAT 999,999,999,999.99 HEADING "Sales";

COLUMN diffPrevYear FORMAT 999,999,999,999.99 HEADING "Difference (Amount)";


TTITLE LEFT "Annual Sales Report" SKIP 1


CREATE OR REPLACE VIEW yearly_sales AS

SELECT year, SUM(totalAmount) AS totalSAmount

FROM sales_fact SF

JOIN date_dim D ON SF.dateKey = D.date_key

WHERE year BETWEEN 2019 AND 2023

GROUP BY year

ORDER BY year;


SELECT YS.year AS "Year",

      TO_CHAR(YS.totalSAmount, '999,999,999,999.99') AS "Sales",

     TO_CHAR(YS.totalSAmount - LAG(YS.totalSAmount) OVER (ORDER BY YS.year),
     '999,999,999,999.99') AS "Difference"


FROM yearly_sales YS

ORDER BY YS.year;

spool off
```

Output:

```
Annual Sales Report

 Year Sales               Difference(Amount)

----- ------------------- -------------------

 2019    1,321,235,955.68

 2020    1,327,717,677.48       6,481,721.80

 2021    1,327,535,339.58         -182,337.90

 2022    1,330,218,582.54        2,683,242.96
 2023      111,542,095.36   -1,218,676,487.18
```

3.4.3 Top 5 total sales states in 2022

This report is to identify the top 5 total sales states in a specific year which is in the year 2022. The total sales amount encompasses the revenue generated from the primary business operations of the company. The objective of this report is to ascertain the states that have excelled in terms of monthly sales performance. This report's structure will be organizing the columns in the order of Ranking, State, Sales Amount. This report equips management with the insights needed to recognize the contributing factors to success and to acknowledge the states that have made the most substantial contributions to the overall sales performance.

```
SET PAGESIZE 35
SET LINESIZE 130
ACCEPT year NUMBER FORMAT '9999' PROMPT 'Enter the Year (YYYY): '

COLUMN RANK FORMAT '99' HEADING "Rank"
COLUMN state FORMAT A20 HEADING "State"
COLUMN sales FORMAT '999,999,999,999,999,999,999.00' HEADING "Sales Amount"

TTITLE CENTER 'The Top 5 Total Sales State In ' &year SKIP 2

CREATE OR REPLACE VIEW topState AS
WITH yearly_sales AS (
  SELECT
    C.state,
    SUM(SF.totalAmount) AS totalSAmount
  FROM sales_fact SF
    JOIN date_dim D ON SF.dateKey = D.date_key
    JOIN customer_dim C ON SF.customerKey = C.customerKey
    JOIN product_dim P ON SF.productKey = P.productKey
  WHERE D.year = &year
  GROUP BY C.state
),
ranking AS (
  SELECT
    DENSE_RANK() OVER (ORDER BY totalSAmount DESC) AS rank,
    state,
    totalSAmount
  FROM yearly_sales
)
SELECT
  rank,
  state,
  totalSAmount AS sales
FROM ranking
WHERE rank <= 5
ORDER BY rank ASC;
```

```
SELECT *
FROM topState
ORDER BY sales DESC;

spool off
```

Output:

```
The Top 5 Total Sales State In 2022

Rank State                                  Sales Amount
---- -------------------- ------------------------------
   1 Selangor                         530,402,829,499.00
   2 Pulau Pinang                     293,826,717,322.00
   3 Pahang                           198,225,685,735.00
   4 Sarawak                          193,833,433,782.00
   5 Johor                            146,252,257,445.00
```