TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

BAIT3003 DATA WAREHOUSE

2022/2023

Student's name/ ID Number	:	Tan Jacqueline 21WMR03259
Student's name/ ID Number	:	Sit Yie Sian 21WMR03693
Student's name/ ID Number	:	Leong Sheng Mou 21WMR07568
Student's name/ ID Number	:	Loke Tze Min 21WMR07394
Tutorial Group	:	RDS2S3G2
Tutor's name	:	Mr Choong Yun Loong

Group Member	Task 3 marks	Total marks
1.Tan Jacqueline	( )	( )
2.Sit Yie Sian	( )	( )
3.Leong Sheng Mou	( )	( )
4.Loke Tze Min	( )	( )

BAIT3003 Data Warehouse Technology
Assignment Assessment Form

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

Table Of Contents

Chapter 1 Design of Data Warehouse

Introduction

The database is based on a global fictitious company that sells computer hardware including storage, motherboard, RAM, video card, and CPU.

The company maintains the product information such as name, description standard cost, list price, and product line. It also tracks the inventory information for all products including warehouses where products are available. Because the company operates globally, it has warehouses in various locations around the world.

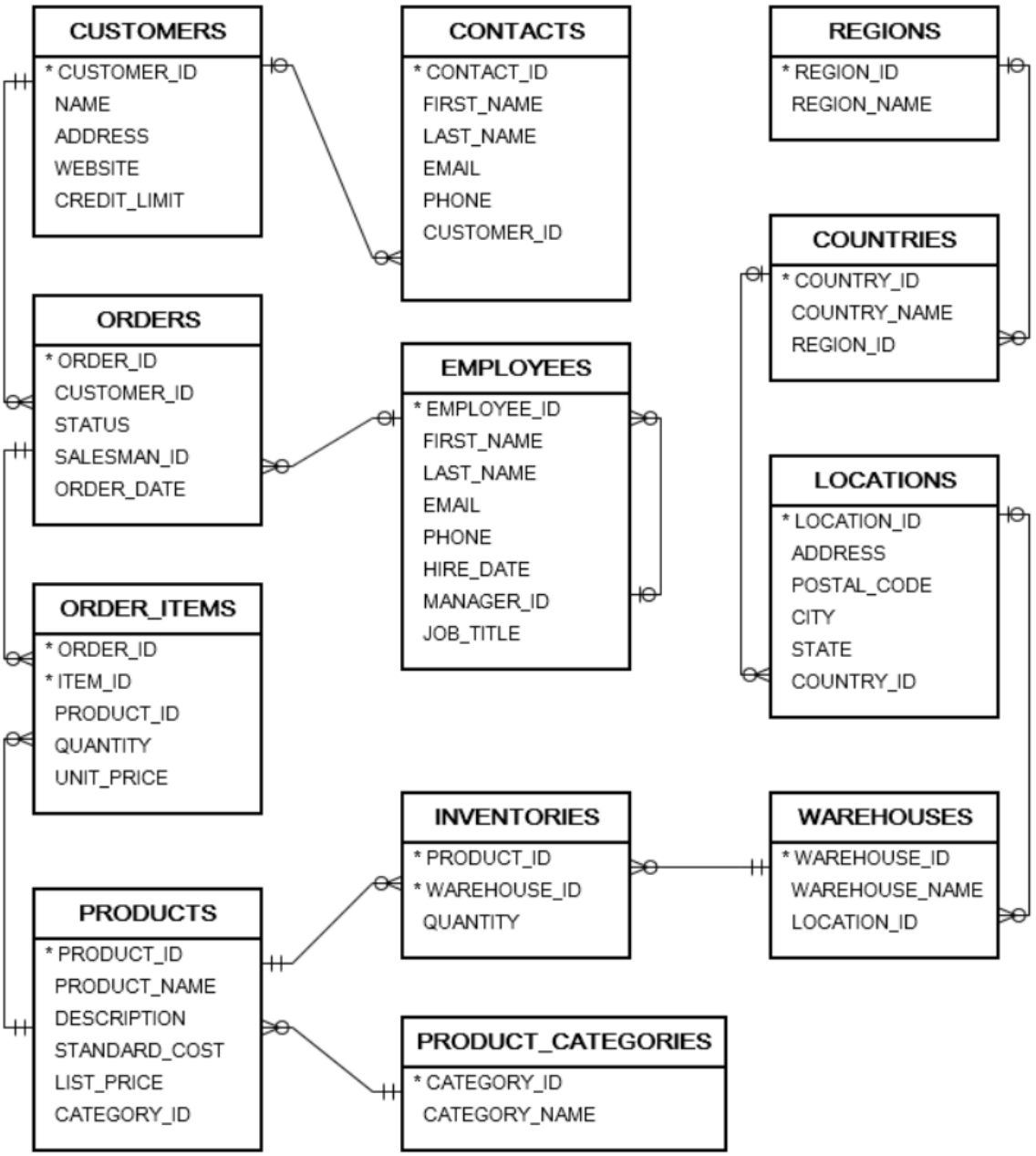
The company records all customer information including name, address, and website. Each customer has at least one contact person with detailed information including name, email, and phone. The company also places a credit limit on each customer to limit the amount that customer can owe.

Whenever a customer issues a purchase order, a sales order is created in the database with the pending status. When the company ships the order, the order status becomes shipped. In case the customer cancels an order, the order status becomes canceled.

In addition to the sales information, the employee data is recorded with some basic information such as name, email, phone, job title, manager, and hire date.

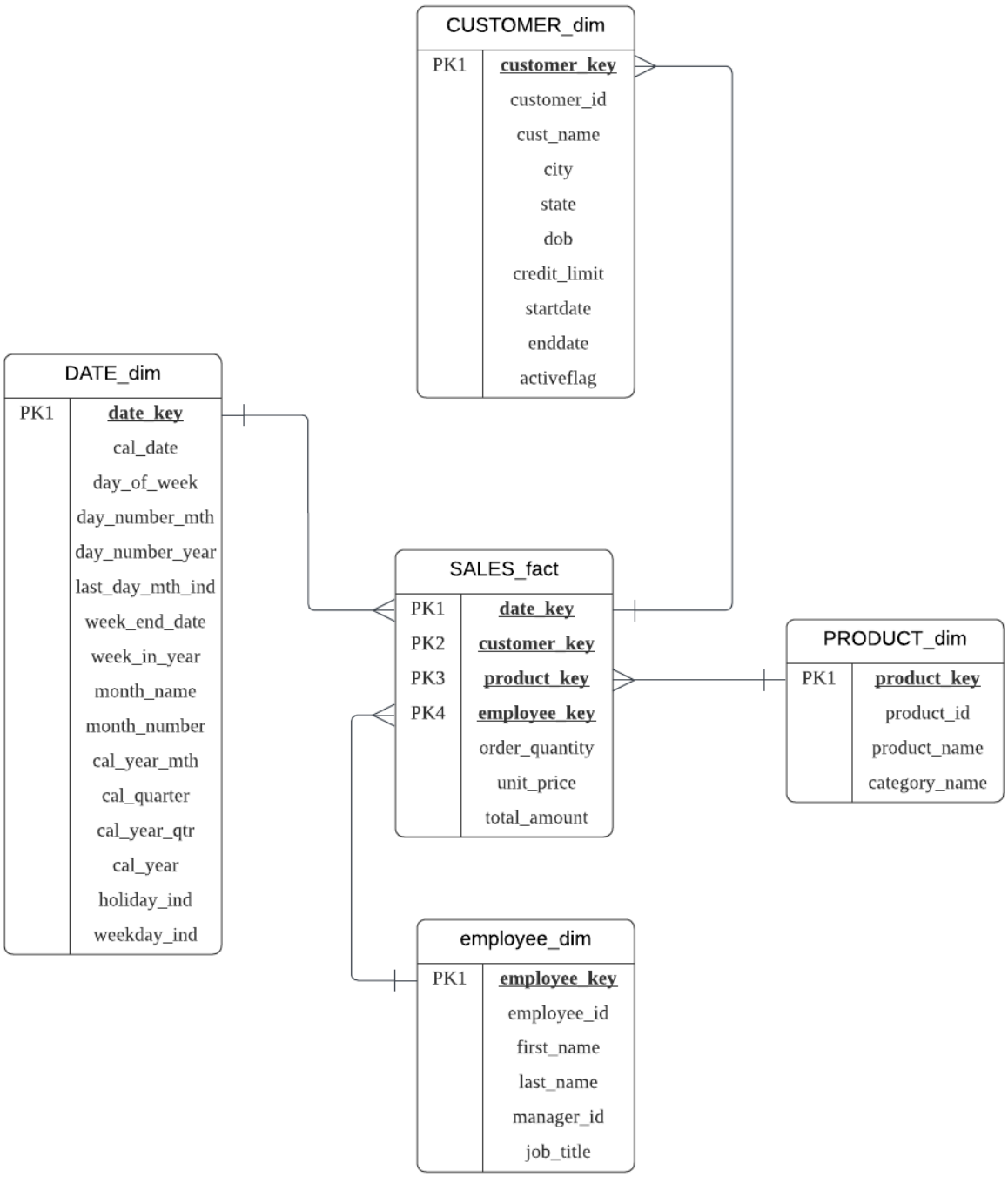
1.1 Original Database

1.1.1 Logical Design



1.2 Star Schema Dimension and Fact Tables

1.2.1 Logical Design



1.2.2 Physical Design

1.2.2.1 Dimension Tables

```
--date_dim
CREATE TABLE DATE_dim (
 date_key         number not null, -- running number
 cal_date         date,            -- e.g. '09/03/2023'
 day_of_week      number(1),       -- 1 to 7
 day_number_mth   number(2),       -- 1 to 31
 day_number_year  number(3),       -- 1 to 366
 last_day_mth_ind char(1),         -- 'Y/N'
 week_end_date    date,            -- date of the week end
 week_in_year     number(2),       -- 1 to 52/53
 month_name       varchar(9),      -- January to december
 month_number     number(2),       -- 1 to 12
 cal_year_mth     char(7),         -- e.g. 2023-03
 cal_quarter      char(2),         -- e.g. q1-q4
 cal_year_qtr     char(6),         -- e.g. 2023Q1
 cal_year         number(4),       -- e.g. 2023
 holiday_ind      char(1),         -- 'Y/N'
 weekday_ind      char(1)          -- 'Y/N'
);

-- dim_product
CREATE TABLE Product_dim
(product_key   number not null,
 product_id    number not null,
 product_name  varchar(50),
 category_name varchar(50),
primary key(product_key)
);

-- dim_customer
CREATE TABLE CUSTOMER_dim
(customer_key    number not null,
 customer_id     number not null,
 cust_name       varchar(30),
 city            varchar(30),
 state           varchar(30),
 credit_limit    number(8,2)
);

-- dim_employee
CREATE TABLE employee_dim
(employee_key   number   not null,
 employee_id    number   not null,
 First_name   varchar(50)      not null,
 Last_name    varchar(50)      not null,
 Manager_id    number                 ,
 Job_title     varchar(50)    not null,
primary key(employee_key)
);
```

1.2.2.2 Fact Table

```
CREATE TABLE SALES_fact
(date_key   number not null,
 customer_key      number not null,
 product_key       number not null,
 employee_key      number not null,
 order_quantity    number(3) not null,
 unit_price number(8,2) not null,
 total_amount      number(11,2) not null,
primary key(date_key, customer_key,
         product_key, employee_key)
);
```

1.3.3 Data Dictionary

Date Dimension

Attribute	Data Type	Constraint	Example
date_key	NUMBER	NOT NULL	11882, 11883
cal_date	DATE	-	25-FEB-23
day_of_week	NUMBER(1)	-	1 to 7
day_number_mth	NUMBER(2)	-	1 to 31
day_number_year	NUMBER(3)	-	1 to 365
last_day_mth_ind	CHAR(1)	-	Y or N
week_end_date	DATE	-	31-DEC-22, 07-JAN-23
week_in _year	NUMBER(2)	-	1 to 52
month_name	VARCHAR(9)	-	January to December
month_number	NUMBER(2)	-	1 to 12
cal_year_mth	CHAR(7)	-	2023-01, 2022-12
cal_quarter	CHAR(2)	-	Q1 to Q4
cal_year_qtr	CHAR(6)	-	2022Q4, 2023Q1
cal_year	NUMBER(4)	-	2022, 2023
holiday_ind	CHAR(1)	-	Y or N
weekday_ind	CHAR(1)	-	Y or N

Customer Dimension

Attribute	Data Type	Constraint	Example
customer_key	NUMBER	NOT NULL	30415, 30417
customer_id	NUMBER	NOT NULL	120415, 120413
cust_name	VARCHAR2(30)	-	BB&T Corp, Raytheon
city	VARCHAR2(30)	-	Sepang, Kuching
state	VARCHAR2(30)	-	Seremban, Sarawak
dob	DATE	-	21-JUL-87, 16SEP-71
credit_limit	NUMBER(8,2)	-	30382, 30231
startdate	DATE	NOT NULL	01-JAN-17
enddate	DATE	NOT NULL	31-DEC-99
activeflag	CHAR(1)	NOT NULL	Y or N

Product Dimension

Attribute	Data Type	Constraint	Example
product_key	NUMBER	NOT NULL	1001, 1288
product_id	NUMBER	NOT NULL	1, 288
product_name	VARCHAR2(50)	-	Seagate ST1000DM010
category_name	VARCHAR2(50)	-	Mother Board, Storage

Employee Dimension

Attribute	Data Type	Constraint	Example
employee_key	NUMBER	NOT NULL	1, 107
employee_id	NUMBER	NOT NULL	1, 107
first_name	VARCHAR(50)	NOT NULL	Harper, Gracie
last_name	VARCHAR(50)	NOT NULL	Spencer, Gardner
maanger_id	NUMBER	-	1, 106
job_title	VARCHAR(50)	NOT NULL	Public Accountant, President

Sales Fact

Attribute	Data Type	Constraint	Example
date_key	NUMBER	NOT NULL	11882, 11883
customer_key	NUMBER	NOT NULL	30415, 30417
product_key	NUMBER	NOT NULL	1001, 1288
employee_key	NUMBER	NOT NULL	1, 107
order_quantity	NUMBER(3)	NOT NULL	1, 5
unit_price	NUMBER(8,2)	NOT NULL	26.99, 299.89
total_amount	NUMBER(11,2)	NOT NULL	9029.85, 899.99

Chapter 2 Extract, Transform, Load Process

2.1 Script for Initial Loading

2.1.1 Date Dimension

```
drop sequence date_seq;
create sequence date_seq
start with 10001
increment by 1;

drop table DATE_dim;
create table DATE_dim (
 date_key         number not null, -- running number
 cal_date         date,            -- e.g. '09/03/2023'
 day_of_week      number(1),       -- 1 to 7
 day_number_mth   number(2),       -- 1 to 31
 day_number_year  number(3),       -- 1 to 366
 last_day_mth_ind char(1),         -- 'Y/N'
 week_end_date    date,            -- date of the week end
 week_in_year     number(2),       -- 1 to 52/53
 month_name       varchar(9),      -- January to december
 month_number     number(2),       -- 1 to 12
 cal_year_mth     char(7),         -- e.g. 2023-03
 cal_quarter      char(2),         -- e.g. q1-q4
 cal_year_qtr     char(6),         -- e.g. 2023Q1
 cal_year         number(4),       -- e.g. 2023
 holiday_ind      char(1),         -- 'Y/N'
 weekday_ind      char(1)          -- 'Y/N'
);


delete Date_dim;
declare
   v_startDate date:=to_date('01/01/2018', 'dd/mm/yyyy');
   v_endDate   date:=to_date('01/03/2023', 'dd/mm/yyyy');

   v_day_of_week      number(1);
   v_day_number_mth   number(2);
   v_day_number_year  number(3);
   v_last_day_mth_ind char(1);
   v_week_end_date    date;
   v_week_in_year     number(2);
   v_month_name       varchar(9);
   v_month_number     number(2);
   v_cal_year_mth     char(7);
   v_cal_quarter      char(2);
   v_cal_year_qtr     char(6);
   v_cal_year         number(4);
   v_holiday_ind      char(1);
   v_weekday_ind      char(1);
```

```
begin
   v_holiday_ind :='N'; --default value
   while(v_startDate<=v_endDate) loop
         v_day_of_week      := to_char(v_startDate, 'd');
         v_day_number_mth   := to_char(v_startDate, 'dd');
         v_day_number_year  := to_char(v_startDate, 'ddd');
         v_week_in_year     := to_char(v_startDate, 'IW');
         v_month_name       := to_char(v_startDate, 'Month');
         v_month_number     := to_char(v_startDate, 'MM');
         v_cal_year_mth     := to_char(v_startDate, 'YYYY-MM');
         v_cal_year         := to_char(v_startDate, 'YYYY');

         v_week_end_date    := v_startDate+(7-v_day_of_week);

         if(v_startDate=last_day(v_startDate)) then
            v_last_day_mth_ind := 'Y';
         else
            v_last_day_mth_ind := 'N';
         end if;

         if(v_day_of_week BETWEEN 2 AND 6) then
            v_weekday_ind := 'Y';
         else
            v_weekday_ind := 'N';
         end if;

         if(v_month_number <=3) then
            v_cal_quarter := 'Q1';
         elsif(v_month_number <=6) then
            v_cal_quarter := 'Q2';
         elsif(v_month_number <=9) then
            v_cal_quarter := 'Q3';
         else
            v_cal_quarter := 'Q4';
         end if;

         v_cal_year_qtr     := v_cal_year || v_cal_quarter;


         insert into DATE_dim values(
               date_seq.nextval,
               v_startDate,
                   v_day_of_week,
                   v_day_number_mth,
                   v_day_number_year,
                   v_last_day_mth_ind,
                   v_week_end_date,
                   v_week_in_year,
                   v_month_name,
                   v_month_number,
                   v_cal_year_mth,
                   v_cal_quarter,
                   v_cal_year_qtr,
                   v_cal_year,
                   v_holiday_ind,
                   v_weekday_ind
           );
```

```
      v_startDate:=v_startDate+1;
   end loop;
end;
/
SELECT COUNT(*) FROM DATE_DIM;
```

2.1.2 Customer Dimension

```
DROP SEQUENCE customer_seq;
CREATE SEQUENCE customer_seq
START WITH 10001
INCREMENT BY 1;

drop table customer_dim;
create table CUSTOMER_dim
(customer_key    number not null,
 customer_id     number not null,
 cust_name       varchar(30),
 city            varchar(30),
 state           varchar(30),
 credit_limit    number(8,2)
);

INSERT INTO customer_dim
SELECT customer_seq.nextvale, customer_id,
               name, city, state, credit_limit
FROM customers;
```

2.1.3 Product Dimension

```
drop sequence prod_dim_seq;
create sequence prod_dim_seq
start with 1001
increment by 1;

drop table Product_dim;

create table Product_dim
(product_key   number not null,
 product_id    number not null,
 product_name  varchar(50),
 category_name varchar(50),
primary key(product_key)
);

insert into product_dim
   select prod_dim_seq.nextval,
          A.product_id,
          substr(A.product_name,1,50),
          substr(B.category_name,1,50)
   from products A
   join product_categories B
      on A.category_id=B.category_id;
```

2.1.4 Employee Dimension

```
drop sequence employee_dim_seq;
create sequence employee_dim_seq
start with 001
increment by 1;

drop table employee_dim;

create table employee_dim
(employee_key   number   not null,
 employee_id    number   not null,
 First_name   varchar(50)      not null,
 Last_name    varchar(50)      not null,
 Manager_id    number                  ,
 Job_title     varchar(50)     not null,
primary key(employee_key)
);

insert into employee_dim
select employee_dim_seq.nextval,
       employee_id,
       First_name,
       Last_name,
       Manager_id,
      Job_title
from employees;

Select count(*) from employee_dim;
```

2.1.5 Sales Fact

```
drop table SALES_fact;
create table SALES_fact
(date_key   number not null,
 customer_key     number not null,
 product_key      number not null,
 employee_key     number not null,
 order_quantity   number(3) not null,
 unit_price number(8,2) not null,
 total_amount     number(11,2) not null,
primary key(date_key, customer_key, product_key, employee_key)
);

insert into SALES_fact
select
      C.date_key,
      D.customer_key,
      E.product_key,
      F.employee_key,
      B.quantity,
      B.unit_price,
      (B.quantity * B.unit_price) total_amount
from orders A
 join order_items B on A.order_id = B.order_id
 join date_dim C on A.order_date = C.cal_date
 join customer_dim D on A.customer_id = D.customer_id
```

```
 join product_dim E on B.product_id = E.product_id
 Join employee_dim F on A.salesman_id = F.employee_id;
```

2.2 Script for Subsequent Loading

2.2.1 Date Dimension

```
ALTER SESSION SET NLS_DATE_FORMAT='DD/MM/YYYY';
exec proc_gen_date('01/01/1970','31/12/2003');
```

2.2.2 Customer Dimension

Add Date of Birth(DOB) for customer Dimension

```
drop sequence gen_date_seq;
create sequence gen_date_seq
start with 10001
increment by 1;

drop table Gen_date;
create table Gen_date
(row_id number,
 row_date date);
delete gen_date;

create or replace procedure proc_gen_date(in_startDate IN varchar,
                                          in_endDate  IN varchar) is

   v_startDate date;
   v_endDate   date;
counter number:=0;
begin
   v_startDate:=to_date(in_startDate,'dd/mm/yyyy');
   v_endDate:=to_date(in_endDate,'dd/mm/yyyy');

   while (v_startDate<=v_endDate) loop
      counter:=counter+1;
      insert into Gen_date values(counter,v_startDate);
      v_startDate:=v_startDate+1;
   end loop;
end;
/

drop table new_cust;
create table new_cust
(c_id       number,
 c_name     varchar(50),
 c_city     varchar(30),
 c_state          varchar(30),
 c_DOB      date,
 c_creditLimit   NUMBER(8,2),
primary key(c_id)
```

```
);

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

delete new_cust;

declare
   cursor cust_cur is
      select * from temp_cust
      order by customer_id;

   t_rec cust_cur%ROWTYPE;

   v_state  varchar(30);
   v_city   varchar(30);
   v_DOB    date;
   v_row_id number;
   v_set    number;
   v_ceditLimit number(8,2);


begin
   for t_rec IN cust_cur loop

      v_set:=trunc(DBMS_RANDOM.value(10001,10042));
      select city, state INTO v_city, v_state
      from StateAndCity
      where StateAndCityID=
            trunc(DBMS_RANDOM.value(v_set,10042));

      v_row_id:=TRUNC(dbms_random.value(1,12419));
      select row_date into v_DOB
      from gen_date
      where row_id= v_row_id;


      insert into new_cust values(
             cust_seq.nextval,
             t_rec.NAME,
             v_city,v_state,
             v_DOB, v_ceditLimit
             );
   end loop;
end;
/

drop table customer_dim;
create table CUSTOMER_dim
(customer_key    number not null,
 customer_id     number not null,
 cust_name       varchar(30),
 city            varchar(30),
 state           varchar(30),
 DOB             date,
```

```
 credit_limit    number(8,2)
);

drop sequence cust_dim_seq;
create sequence cust_dim_seq
start with 10001
increment by 1;

insert into CUSTOMER_dim
select cust_dim_seq.nextval,
       C_ID,
       substr(C_name,1,30),
       C_city,
       C_state,
       C_DOB,
       C_creditLimit
from new_cust;
```

Slowly Changing Dimension Type 2

```
ALTER SESSION SET NLS_DATE_FORMAT ='dd-mm-yyyy';
ALTER TABLE CUSTOMER_dim
ADD startDate date DEFAULT '01-01-2017' NOT NULL;
ALTER TABLE CUSTOMER_dim
ADD endDate date DEFAULT '31-12-9999' NOT NULL;
ALTER TABLE CUSTOMER_dim
ADD activeFlag char(1) DEFAULT 'Y' NOT NULL;

CREATE OR REPLACE PROCEDURE Update_Cust(IN_customerid IN NUMBER,
IN_startDate IN DATE) IS

CURSOR cust_cur is
SELECT customer_id,
       cust_name,
       city,
       state,
       DOB,
       credit_limit
from customer_dim
where customer_id = IN_customerid;

cust_rec cust_cur%ROWTYPE;

BEGIN
    OPEN cust_cur;
    LOOP
      FETCH cust_cur INTO cust_rec;
      EXIT WHEN cust_cur%NOTFOUND;

      Update customer_dim SET endDate = TO_DATE(IN_startDate,'dd-mm-yyyy' )- 1,
activeFlag = 'N'
        WHERE customer_id = IN_customerid;

      INSERT INTO customer_dim values (cust_dim_seq.nextval,
                                IN_customerid,
```

```
                                cust_rec.cust_name,
                                cust_rec.city,
                                cust_rec.state,
                                cust_rec.DOB,
                                cust_rec.credit_limit,
                                IN_startDate,
                                TO_DATE('31-12-9999', 'DD-MM-YYYY'), --
Corrected date format
                                'Y');

     DBMS_OUTPUT.PUT_LINE('customer_id:' || cust_rec.customer_id || '
startDate:' || IN_startDate);
     END LOOP;
     CLOSE cust_cur;
END;
/

exec Update_Cust(100007, '27-04-2023');
select * from customer_dim where customer_id = '100007';
```

Chapter 3 Business Analytics Reports

3.1 Leong Sheng Mou

3.1.1 Report I compare sales weekend and weekdays in 2022

The given code retrieves the monthly sales data for the year 2022, categorized by weekdays and weekends. The data is computed by summing the product of unit price and quantity for each order item, for both weekdays and weekends. The view also includes the total sales for each month. The code provides useful insights into the sales trends during weekdays and weekends, which can be used to make informed decisions about marketing campaigns, promotions, and inventory management. By comparing the sales figures for weekdays and weekends, businesses can determine whether there are any significant differences in consumer behavior during these two periods, and adjust their strategies accordingly.

SQL Query:

```
set linesize 120

set pagesize 35


CREATE OR REPLACE VIEW monthly_sales_2022 AS

     SELECT

       CASE EXTRACT(MONTH FROM d.CAL_DATE)

         WHEN 1 THEN 'January'

         WHEN 2 THEN 'February'

         WHEN 3 THEN 'March'

         WHEN 4 THEN 'April'

         WHEN 5 THEN 'May'

         WHEN 6 THEN 'June'

         WHEN 7 THEN 'July'

         WHEN 8 THEN 'August'
```

```
          WHEN 9 THEN 'September'

          WHEN 10 THEN 'October'

          WHEN 11 THEN 'November'

          WHEN 12 THEN 'December'

        END AS MONTH,

        SUM(CASE WHEN d.WEEKDAY_IND = 'Y' THEN oi.UNIT_PRICE * oi.QUANTITY ELSE
0 END) AS WEEKDAY_SALES,

        SUM(CASE WHEN d.WEEKDAY_IND = 'N' THEN oi.UNIT_PRICE * oi.QUANTITY ELSE
0 END) AS WEEKEND_SALES,

        SUM(CASE WHEN d.WEEKDAY_IND = 'Y' THEN oi.UNIT_PRICE * oi.QUANTITY ELSE
- oi.UNIT_PRICE * oi.QUANTITY END) AS WEEKDAY_DIFF,

        SUM(CASE WHEN d.WEEKDAY_IND = 'N' THEN oi.UNIT_PRICE * oi.QUANTITY ELSE
- oi.UNIT_PRICE * oi.QUANTITY END) AS WEEKEND_DIFF,

        SUM(oi.UNIT_PRICE * oi.QUANTITY) AS TOTAL_SALES

     FROM orders o

     JOIN order_items oi ON o.ORDER_ID = oi.ORDER_ID

     JOIN customer_dim c ON o.CUSTOMER_ID = c.CUSTOMER_ID

     JOIN date_dim d ON o.ORDER_DATE = d.CAL_DATE

     WHERE EXTRACT(YEAR FROM d.CAL_DATE) = 2022

     GROUP BY EXTRACT(MONTH FROM d.CAL_DATE)

     ORDER BY EXTRACT(MONTH FROM d.CAL_DATE);


View created.

SELECT * FROM monthly_sales_2022;

spool off
```

Sun Apr 30
page 1

Monthly Sales Weekday and Weekend 2022

MONTH	WEEKDAY_SALES	WEEKEND_SALES	WEEKDAY_DIFF	WEEKEND_DIFF	Total Sales
January	26502289.5	12964448.8	13537840.7	-13537841	$39,466,738
February	26263906.7	8072176.48	18191730.2	-18191730	$34,336,083
March	25634873.9	9919265.51	15715608.4	-15715608	$35,554,139
April	22645907.9	10523976.8	12121931.2	-12121931	$33,169,885
May	24534531.3	8390426.45	16144104.8	-16144105	$32,924,958
June	24912528.3	8144091.6	16768436.7	-16768437	$33,056,620
July	24828486.7	10890062.7	13938424	-13938424	$35,718,549
August	28588542.1	10276063.7	18312478.3	-18312478	$38,864,606
September	27655492.3	9575961.27	18079531.1	-18079531	$37,231,454
October	23601985.2	13794362.4	9807622.84	-9807622.8	$37,396,348
November	25758596.3	9850820.98	15907775.3	-15907775	$35,609,417
December	22311412	9839845.98	12471566.1	-12471566	$32,151,258

12 rows selected.

The given table shows the monthly sales data for both weekdays and weekends in the year 2022. The table contains 12 rows, one for each month, and displays the weekday sales, weekend sales, the difference between weekday and weekend sales, and the total sales for each month. The sales data suggests that the sales figures are generally higher on weekdays than on weekends, with a few exceptions. The total sales for the year 2022 is $377,264,908. Overall, this data can be used by businesses to analyze their sales patterns and make informed decisions about their operations, such as adjusting staffing levels or changing promotional strategies.

3.1.2 Report 2 list all total spend amount based on city 2021

The following report provides a comprehensive analysis of the total spend amount based on the city in Malaysia for the year 2021. The report aims to provide insights into the spending habits of customers in different cities in Malaysia, and identify any trends or patterns that may exist. The report provides a breakdown of the total spend amount for each city, along with any key observations or findings that may be of interest. This information can be useful for businesses and organizations operating in Malaysia to understand consumer behavior and optimize their marketing and sales strategies accordingly.

SQL Query:

```
 set linesize 120

 set pagesize 35

 CREATE OR REPLACE VIEW city_spend_2021_rank_diff AS

     SELECT c.CITY,

     SUM(oi.QUANTITY * oi.UNIT_PRICE) AS TOTAL_SPEND,

     ROUND(SUM(oi.QUANTITY * oi.UNIT_PRICE) * 100 / SUM(SUM(oi.QUANTITY *
oi.UNIT_PRICE)) OVER (), 2) AS PERCENTAGE_OF_TOTAL_SPEND,

     RANK() OVER (ORDER BY SUM(oi.QUANTITY * oi.UNIT_PRICE) DESC) AS
SPEND_RANK,

     COALESCE(SUM(oi.QUANTITY * oi.UNIT_PRICE) - LAG(SUM(oi.QUANTITY *
oi.UNIT_PRICE)) OVER (ORDER BY SUM(oi.QUANTITY * oi.UNIT_PRICE) DESC), 0) AS
SPEND_DIFF

     FROM orders o

     JOIN order_items oi ON o.ORDER_ID = oi.ORDER_ID

     JOIN customer_dim c ON o.CUSTOMER_ID = c.CUSTOMER_ID

      JOIN date_dim d ON o.ORDER_DATE = d.CAL_DATE

      WHERE d.CAL_YEAR = 2021

      GROUP BY c.CITY;

View created.

 SELECT * FROM city_spend_2021_rank_diff;
```

```
spool off
```

City Spend 2021 Rank and Difference

CITY	TOTAL_SPEND	PERCENTAGE_OF_TOTAL_SPEND	SPEND_RANK	SPEND_DIFF
Kuala Lumpur	45170628.2	10.66	1	0
Cheras	34940515.9	8.24	2	-10230112
Subang Jaya	30182063.8	7.12	3	-4758452.1
Shah Alam	26417028.9	6.23	4	-3765034.9
Seri Kembangan	24638562	5.81	5	-1778466.9
Sepang	20692052.4	4.88	6	-3946509.7
Rawang	17903302.2	4.22	7	-2788750.2
Petaling Jaya	17000115.4	4.01	8	-903186.79
Klang	16843830.4	3.97	9	-156285.05
Kajang	15593261.2	3.68	10	-1250569.2
Banting	14547961.2	3.43	11	-1045299.9
Balakong	12661029.9	2.99	12	-1886931.3
Ampang	12541340.5	2.96	13	-119689.44
Sibu	11685407.2	2.76	14	-855933.33

Miri	10298093.8	2.43	15	-1387313.4
Kuching	10138401.8	2.39	16	-159692.03
Kapit	9148922.9	2.16	17	-989478.89
Tawau	8761399.21	2.07	18	-387523.69
Kota Kinabalu	7816755.92	1.84	19	-944643.29
Seberang Perai	7647825.87	1.8	20	-168930.05
Perai	7195468.25	1.7	21	-452357.62
Pulau Tikus	6997400.03	1.65	22	-198068.22
GeorgeTown	6045687.52	1.43	23	-951712.51
Butterworth	5948506.67	1.4	24	-97180.85
Bukit Mertajam	5048203.23	1.19	25	-900303.44
Kuala Perlis	4948530.43	1.17	26	-99672.8
Bidor	4536591.68	1.07	27	-411938.75
Ipoh	4125822.01	.97	28	-410769.67
Kuantan	3651357.82	.86	29	-474464.19

City Spend 2021 Rank and Difference

CITY	TOTAL_SPEND	PERCENTAGE_OF_TOTAL_SPEND	SPEND_RANK	SPEND_DIFF
Mentakab	3599750.43	.85	30	-51607.39
Kuala Lipis	2539114.46	.6	31	-1060636
Seremban	2487114.17	.59	32	-52000.29
Port Dickson	2379824.62	.56	33	-107289.55
Genting Highlands	2292367.28	.54	34	-87457.34
Melaka	1770468.35	.42	35	-521898.93
Bunut Payong	1649876.54	.39	36	-120591.81
Alor Setar	1281288.27	.3	37	-368588.27
Langkawi	1209740.98	.29	38	-71547.29
Kota Tinggi	706083.72	.17	39	-503657.26
Kluang	554953.91	.13	40	-151129.81
Johor Bahru	182370.36	.04	41	-372583.55

41 rows selected.

The report presents a comprehensive list of the total spend amount based on different cities in Malaysia for the year 2021. Kuala Lumpur tops the list with a total spend of over 45 million, accounting for 10.66% of the total spend. Cheras, Subang Jaya, and Shah Alam follow closely in terms of total spend, while Johor Bahru has the lowest spend with only 182,370.36, accounting for just 0.04% of the total spend. The report also includes the spend rank and the spend difference of each city. This information can be used by businesses to make strategic decisions related to investment and expansion in different cities.

3.1.3 Report 3 compare top 3 product from 2020 to 2023

Based on the sales rank view for the top three products in 2020 and 2023, there have been some changes in their rankings. The report provides valuable insights into changes in customer preferences over time and can help businesses make informed decisions regarding their product offerings.

SQL Query:

```
set linesize 120

set pagesize 35

COLUMN PRODUCT_NAME HEADING 'Product Name'

COLUMN CAL_YEAR HEADING 'Year'

COLUMN TOTAL_SALES FORMAT $99,999,999 HEADING 'Total Sales'

COLUMN SALES_DIFFERENCE FORMAT $99,999,999 HEADING 'Sales Difference'

BREAK ON CAL_YEAR SKIP 1


CREATE OR REPLACE VIEW top3_product_sales_rank_view AS

    WITH top3_products AS (

        SELECT

            prod.PRODUCT_NAME,

            SUM(oi.QUANTITY * oi.UNIT_PRICE) AS TOTAL_SALES

        FROM

            orders o

            JOIN order_items oi ON o.ORDER_ID = oi.ORDER_ID

            JOIN product_dim prod ON oi.PRODUCT_ID = prod.PRODUCT_ID

            JOIN date_dim d ON o.ORDER_DATE = d.CAL_DATE

        WHERE

            d.CAL_DATE BETWEEN ADD_MONTHS(TRUNC(SYSDATE, 'YEAR'), -36) AND
TRUNC(SYSDATE)
```

```
        GROUP BY

            prod.PRODUCT_NAME

        ORDER BY

            TOTAL_SALES DESC

    )

    SELECT

        RANK() OVER (PARTITION BY d.CAL_YEAR ORDER BY total_sales DESC) AS
sales_rank,

        prod.PRODUCT_NAME,

        d.CAL_YEAR,

    SUM(oi.QUANTITY * oi.UNIT_PRICE) AS TOTAL_SALES,

        (SUM(oi.QUANTITY * oi.UNIT_PRICE) - LAG(SUM(oi.QUANTITY *
oi.UNIT_PRICE)) OVER (PARTITION BY prod.PRODUCT_NAME ORDER BY d.CAL_YEAR)) AS
SALES_DIFFERENCE

    FROM

        orders o

        JOIN order_items oi ON o.ORDER_ID = oi.ORDER_ID

        JOIN product_dim prod ON oi.PRODUCT_ID = prod.PRODUCT_ID

        JOIN date_dim d ON o.ORDER_DATE = d.CAL_DATE

        JOIN (

            SELECT PRODUCT_NAME, TOTAL_SALES

            FROM top3_products

            WHERE ROWNUM <= 3

    ) t3 ON prod.PRODUCT_NAME = t3.PRODUCT_NAME

    WHERE

        d.CAL_DATE BETWEEN ADD_MONTHS(TRUNC(SYSDATE, 'YEAR'), -36) AND
TRUNC(SYSDATE)

    GROUP BY

        prod.PRODUCT_NAME,

        d.CAL_YEAR,
```

```
         total_sales
      ORDER BY
         d.CAL_YEAR ASC,
         sales_rank;

View created.

 SELECT * FROM top3_product_sales_rank_view ;

spool off
```

City Spend 2021 Rank and Difference

SALES_RANK	Product Name	Year	Total Sales	Sales Difference
1	G.Skill Ripjaws V Series	2020	$17,212,854	
2	Corsair Dominator Platinum		$16,336,317	
3	Intel SSDPECME040T401		$14,215,388	
1	G.Skill Ripjaws V Series	2021	$16,709,259	-$503,595
2	Corsair Dominator Platinum		$15,427,164	-$909,153
3	Intel SSDPECME040T401		$13,381,797	-$833,591
1	G.Skill Ripjaws V Series	2022	$16,628,287	-$80,971
2	Corsair Dominator Platinum		$16,051,558	$624,394
3	Intel SSDPECME040T401		$13,638,969	$257,172
1	G.Skill Ripjaws V Series	2023	$2,804,400	-$13,823,887
2	Corsair Dominator Platinum		$2,802,669	-$13,248,890
3	Intel SSDPECME040T401		$2,740,209	-$10,898,760

12 rows selected

This report displays the sales rank and total sales of the top 3 products in 2020, 2021, 2022, and 2023. In 2020, G.Skill Ripjaws V Series was the top-selling product with total sales of $17,212,854, followed by Corsair Dominator Platinum and Intel SSDPECME040T401. However, in 2021, there was a decrease in sales for all three products. The sales of G.Skill Ripjaws V Series decreased by $503,595, while the sales of Corsair Dominator Platinum decreased by $909,153. The sales of Intel SSDPECME040T401 decreased by $833,591. In 2022, the sales of G.Skill Ripjaws V Series continued to decrease, while the sales of Corsair Dominator Platinum increased by $624,394. In 2023, all three products experienced a significant drop in sales, with G.Skill Ripjaws V Series, Corsair Dominator Platinum, and Intel SSDPECME040T401 having sales differences of -$13,823,887, -$13,248,890, and -$10,898,760, respectively.

3.2 Loke Tze Min

3.2.1 Report 1: Compare the 5 product category of 2021 and 2022

The report will show the existing product category sales comparison for 2021 and 2022. This can help the company to view which product category contributes the highest sales to the business. In the report, the highest sales figure of the product category can be described from 2 perspectives which are the sales figure in currency and also the total amount of product sold in that particular category. The report will also output which product from the top sale product category is the best sales among all the products in the same category. Other than visualizing the top sales category, this report also can help the company to determine which product and product category should increase the inventory stock order for future stock ordering.

SQL Query:

```
SET LINESIZE 200
SET PAGESIZE 50
SET SERVEROUTPUT ON

CREATE OR REPLACE VIEW Total_Sales AS
                   SELECT B.cal_year AS year, C.category_name,
                  SUM(A.order_quantity) AS Qty,
                  SUM(total_amount) AS ProductSale
                  FROM sales_fact A
                  JOIN date_dim B ON A.date_key = B.date_key
                  JOIN product_dim C ON A.product_key = C.product_key
                   GROUP BY B.cal_year, C.category_name
                  ORDER BY ProductSale DESC;

CREATE OR REPLACE PROCEDURE Top_Product_Cat IS

                v_bestProductCatName     VARCHAR(70);
                 v_bestProductName        VARCHAR(70);
                v_bestSales              NUMBER(20,2);
                v_quantity               NUMBER(10);
                v_subTotal               NUMBER(15,2);
                 v_qty2022                NUMBER(5);

CURSOR top5_cur IS
    SELECT *
    FROM (
      SELECT category_name,
             SUM(total_amount) AS ProductSale,
             SUM(order_quantity) AS QTY
      FROM sales_fact A
      JOIN date_dim B ON A.date_key = B.date_key
```

```
      JOIN product_dim C ON A.product_key = C.product_key
      WHERE B.cal_year =2021
      GROUP BY category_name
      ORDER BY SUM(total_amount) DESC
    )
    WHERE ROWNUM <= 5;

CURSOR subTotal_cur IS
       SELECT SUM(total_amount) AS SubTotal
       FROM sales_fact A
       JOIN date_dim B ON A.date_key = B.date_key
       WHERE B.cal_year IN (2021, 2022);

CURSOR bestProdCat_cur IS
       SELECT category_name, product_name, ProductSale
       FROM (
             SELECT C.category_name, C.product_name,
             SUM(A.order_quantity) As Qty ,
             SUM(total_amount)AS ProductSale
             FROM sales_fact A
             JOIN date_dim B ON A.date_key = B.date_key
             JOIN product_dim C ON A.product_key = C.product_key
             WHERE B.cal_year IN (2021, 2022)
             GROUP BY C.category_name, product_name
             ORDER BY ProductSale DESC)

             WHERE RowNum <= 5;


CURSOR bestQty_cur IS
       SELECT order_quantity
       FROM(
        SELECT A.order_quantity
        FROM sales_fact A
        JOIN date_dim B ON A.date_key = B.date_key
        WHERE B.cal_year IN (2021, 2022)
        ORDER BY A.order_quantity DESC)

        WHERE rownum = 1;

BEGIN
             DBMS_OUTPUT.PUT_LINE(LPAD ('=', 98, '='));
             DBMS_OUTPUT.PUT_LINE(' | ' || LPAD ('Sales Comparison by
Category',60,' ')
             || LPAD(' | ', 37));
             DBMS_OUTPUT.PUT_LINE(' | ' || RPAD(' ', 94) || ' | ');
             DBMS_OUTPUT.PUT_LINE(' | ' ||  LPAD('Date Generated: ', 75)
             || TO_CHAR(SYSDATE, 'DD-MM-YYYY HH24:MI:SS') || ' | ');
             DBMS_OUTPUT.PUT_LINE(LPAD('-', 98, '-'));
             DBMS_OUTPUT.PUT_LINE(' | '||RPAD('Product Category',20)||' |
'||('Sales 2021 (RM)')||' | '||('Sales 2022 (RM)')||' | '|| RPAD('Sales QTY
2021 ', 15)||' | '|| RPAD('Sales QTY 2022 ', 17)||' | ');
             DBMS_OUTPUT.PUT_LINE(LPAD ('-',98,'-'));

          FOR SALES_REC IN top5_cur LOOP
            SELECT SUM(total_amount), SUM(order_quantity)
            INTO v_subTotal, v_quantity
```

```
            FROM sales_fact A
            JOIN date_dim B ON A.date_key = B.date_key
            JOIN product_dim C ON A.product_key = C.product_key
            WHERE B.cal_year = 2022
            AND C.category_name = SALES_REC.category_name;
            DBMS_OUTPUT.PUT_LINE(' | '|| RPAD(SALES_REC.category_name,20)||' |
'||
            RPAD(TO_CHAR(SALES_REC.ProductSale,'9,999,99999.99'),15)|| ' | '||
            RPAD(TO_CHAR(v_subtotal,'9,999,99999.99'),15)|| ' | '||
            RPAD(SALES_REC.QTY,15) || ' | '||RPAD(v_quantity,17) || ' | ');
          END LOOP;


      OPEN subTotal_cur;
      FETCH subTotal_cur INTO v_subTotal;

     OPEN bestProdCat_cur;
      FETCH bestProdCat_cur INTO v_bestProductCatName ,v_bestProductName,
v_bestSales;

     OPEN bestQty_cur;
      FETCH subTotal_cur INTO v_qty2022;

            DBMS_OUTPUT.PUT_LINE(LPAD ('-',98,'-'));
            DBMS_OUTPUT.PUT_LINE(' | ' || LPAD('Total Sales (RM): ',70,' ')||
        RPAD(TO_CHAR(v_subTotal,'9,999,9999,99.99'),24, ' ') || ' | ');
            DBMS_OUTPUT.PUT_LINE(' | '|| RPAD(' ',94) ||' | ');

            DBMS_OUTPUT.PUT_LINE(LPAD ('-',40,'-')||'Summary'||LPAD
('-',51,'-'));
            DBMS_OUTPUT.PUT_LINE(' | '||'The Best Sales Product Category:
'||(v_bestProductCatName) || LPAD('|', 56));
            DBMS_OUTPUT.PUT_LINE(' | '||'The Best Sales Product Name: '||
      RPAD(v_bestProductName, 65) ||' | ');

            DBMS_OUTPUT.PUT_LINE(LPAD ('=',98,'='));

            DBMS_OUTPUT.PUT_LINE(LPAD ('-',98, '-'));
            DBMS_OUTPUT.PUT_LINE(' | '||LPAD ('END OF REPORT',50,' ')||LPAD('
| ', 47));
            DBMS_OUTPUT.PUT_LINE(LPAD('=', 98, '='));

END;
/

spool C:\Users\sherm\OneDrive\Desktop\QUERY1_rpt.txt

EXEC Top_Product_Cat;

spool off
```

```
==========================================================================================================
|                                  Sales Comparison by Category                                          |
|                                                                                                        |
|                                                          Date Generated: 30-04-2023 11:58:37           |
----------------------------------------------------------------------------------------------------------
| Product Category     | Sales 2021 (RM) | Sales 2022 (RM) | Sales QTY 2021  | Sales QTY 2022    |
----------------------------------------------------------------------------------------------------------
| CPU                  |  1,620,90372.45 |  1,603,49058.79 | 117148          | 115692            |
| Storage              |  1,161,21536.19 |  1,144,22660.42 | 181580          | 181299            |
| Video Card           |  1,158,57784.33 |  1,154,62904.03 | 82995           | 82836             |
| Mother Board         |    407,12997.74 |    404,36846.57 | 100980          | 100446            |
----------------------------------------------------------------------------------------------------------
|                                                  Total Sales (RM):    865,4541,60.52                   |
|                                                                                                        |
----------------------------------------Summary-------------------------------------------------------
| The Best Sales Product Category: Storage                                                               |
| The Best Sales Product Name: G.Skill Ripjaws V Series                                                  |
==========================================================================================================
----------------------------------------------------------------------------------------------------------
|                                       END OF REPORT                                                    |
==========================================================================================================
```

The diagram above shows the report for comparison of product categories of 2021 and 2022. The date generated of the report will be shown. The report shows the product category name, the sales figure and the sales quantity for 2021 and 2022 respectively. The total sales of both years are also shown as “Total Sales (RM):” which is RM865,4541,60.52 in this case. The report also shows the summary of the best sales product category and best sales product. The best sales product category is evaluated from the sales quantity.

In conclusion, the best sales category is the “Storage” category and the best sales product us “G.Skill Ripjaws V Series”, By referring to the top category sales figure, we can conclude that the top 1 sales figure had decrease by year with value of RM1741313.66

3.2.2 Report 2: Comparing 2021 top 10 customers with 2022 customers, show total spending amount and calculate the difference

This report shows the top 10 customers of 2021 and 2022 and makes a comparison on the sales figure. Besides, it will also show the amount of difference spent by customers. By comparing the amount spent, the company can determine whether the sales increase or decrease and indicate the business plan to overcome why the sales drop in the top 10 category. For example, the sales department of the company can have a site investigation or have a customer meeting to observe the problem and give out suitable promotions or discounts to the royal customer as a reward for being top 10 purchasers from the company.

SQL Query:

```
SET LINESIZE 200
SET PAGESIZE 50
SET SERVEROUTPUT ON


CREATE OR REPLACE VIEW customerPurchase AS
    SELECT *
    FROM (
    SELECT Y.cal_year AS Year,
       C.customer_id as CustomerID,C.cust_name AS CustomerName,
       SUM(total_amount) as TotalSales
    FROM Sales_Fact S
    JOIN customer_dim C
       ON S.customer_key = C.customer_key
    JOIN date_dim Y
      ON S.date_key = Y.date_key
    JOIN product_dim P
      ON S.product_key = P.product_key
    GROUP BY  Y.cal_year ,C.customer_id, C.cust_name
    ORDER BY TotalSales DESC
   );



CREATE OR REPLACE PROCEDURE compareTopCustomer IS
   v_id NUMBER;
   v_name VARCHAR2(50);
   v_sales_2021 NUMBER;
   v_sales_2022 NUMBER;
   v_difference NUMBER;

   CURSOR cur_2021 IS
      SELECT CustomerID, CustomerName, TotalSales
      FROM customerPurchase
```

```
      WHERE RowNum <= 10 AND Year = 2021;

   CURSOR cur_2022 IS
      SELECT CustomerID, CustomerName, TotalSales
      FROM customerPurchase
      WHERE RowNum <= 10 AND Year = 2022;

BEGIN
   DBMS_OUTPUT.PUT_LINE(LPAD ('=', 123, '='));
   DBMS_OUTPUT.PUT_LINE(' | ' || LPAD ('Top 10 Customer Sales Record',80,' ')
                        || LPAD(' | ', 42));
   DBMS_OUTPUT.PUT_LINE(' | ' || RPAD(' ', 119) || ' | ');
   DBMS_OUTPUT.PUT_LINE(' | ' ||  LPAD('Date Generated: ', 100)
            || TO_CHAR(SYSDATE, 'DD-MM-YYYY HH24:MI:SS') || LPAD(' | ', 2));


   DBMS_OUTPUT.PUT_LINE(LPAD('=', 123, '='));
   DBMS_OUTPUT.PUT_LINE('|' || LPAD('Top 10 Customers in 2021', 39) || LPAD('
',13) ||
                        '|' || LPAD('Top 10 Customers in 2022', 40) || LPAD('
',12) ||
                        '|' || LPAD('Difference (RM)', 15) || '|');
   DBMS_OUTPUT.PUT_LINE('|' || LPAD('-', 121, '-') || '|');

   OPEN cur_2021;
   OPEN cur_2022;

   LOOP
      FETCH cur_2021 INTO v_id, v_name, v_sales_2021;
      FETCH cur_2022 INTO v_id, v_name, v_sales_2022;

      EXIT WHEN cur_2021%NOTFOUND OR cur_2022%NOTFOUND;

      v_difference := v_sales_2022 - v_sales_2021;

      DBMS_OUTPUT.PUT_LINE('| ' || LPAD(v_name , 30) ||
                           LPAD(TO_CHAR(v_sales_2021, '9,999,999.99'), 20) || '
| ' ||
                           LPAD(v_name, 30) ||
                           LPAD(TO_CHAR(v_sales_2022, '9,999,999.99'), 20) || '
| ' ||
                           LPAD(TO_CHAR(v_difference, '9,999,999.99'), 13) || '
|');
   END LOOP;

   CLOSE cur_2021;
   CLOSE cur_2022;

   DBMS_OUTPUT.PUT_LINE(LPAD('-', 123, '-'));
   DBMS_OUTPUT.PUT_LINE(' | '||LPAD ('END OF REPORT',70,' ')||LPAD(' | ', 52));
   DBMS_OUTPUT.PUT_LINE(LPAD('=', 123, '='));

END;
/


spool C:\Users\sherm\OneDrive\Desktop\QUERY2_rpt.txt
```

```
EXECUTE compareTopCustomer;

spool off
```

Top 10 Customer Sales Record

Date Generated: 30-04-2023 12:26:49

Top 10 Customers in 2021		Top 10 Customers in 2022		Difference (RM)
Devon Energy	152,230.56	Devon Energy	132,366.90	-19,863.66
United Technologies	129,753.14	United Technologies	125,145.72	-4,607.42
Jabil Circuit	122,390.00	Jabil Circuit	114,079.32	-8,310.68
Intel	119,847.18	Intel	106,651.31	-13,195.87
Delta Air Lines	117,816.05	Delta Air Lines	106,639.89	-11,176.16
CDW	115,677.09	CDW	106,249.09	-9,428.00
Health Net	115,086.54	Health Net	105,802.41	-9,284.13
DaVita HealthCare Partners	115,083.66	DaVita HealthCare Partners	105,575.72	-9,507.94
Core-Mark Holding	114,796.42	Core-Mark Holding	103,721.95	-11,074.47
Fidelity National Financial	112,845.96	Fidelity National Financial	103,331.31	-9,514.65

END OF REPORT

The diagram above shows the report of the top 10 customer sales records for 2021 and 2022. The top 10 customers in both 2021 and 2022 are shown and the sales difference for each top 10 customers are calculated. The report generated date and time is also shown at the top right of the report.

From the printed report, we can observe that both 2021 and 2022 top 10 customers are the same but the overall sales figures are shown as a decrease in each customer.

3.2.3 Report 3: List out the customer who spend over RM110k by year and point out the highest amount for each year

The purpose of this report is to observe the customers who had spent over RM110 thousand each of the year. Besides, the pointer also will highlight who is the top sales contributor of each year. With this report, the company can conclude the top sale contributor and also the sales figure amount respectively.

SQL Query:

```
SET LINESIZE 200
SET PAGESIZE 50
SET SERVEROUTPUT ON

CREATE OR REPLACE VIEW custPurchase AS
    SELECT *
    FROM (
    SELECT Y.cal_year AS Year,
       C.customer_id as CustomerID,C.cust_name AS CustomerName,
       SUM(total_amount) as TotalSales
    FROM Sales_Fact S
    JOIN customer_dim C
       ON S.customer_key = C.customer_key
   JOIN date_dim Y
      ON S.date_key = Y.date_key
   JOIN product_dim P
      ON S.product_key = P.product_key
   GROUP BY  Y.cal_year ,C.customer_id, C.cust_name
   ORDER BY TotalSales DESC
   );

CREATE OR REPLACE PROCEDURE Over110k IS
   v_customer_id custPurchase.CustomerID%TYPE;
   v_customer_name custPurchase.CustomerName%TYPE;
   v_year custPurchase.Year%TYPE;
   v_total_sales custPurchase.TotalSales%TYPE;
   v_max_total_sales custPurchase.TotalSales%TYPE := 0;
   v_max_year custPurchase.Year%TYPE := 0;

BEGIN
   DBMS_OUTPUT.PUT_LINE(LPAD('=', 99, '='));
   DBMS_OUTPUT.PUT_LINE('|' || LPAD('Customers Spending Over RM110k by Year',
70) ||LPAD(' ',27)|| '|');
   DBMS_OUTPUT.PUT_LINE(' | ' || RPAD(' ', 95) || ' | ');
   DBMS_OUTPUT.PUT_LINE(' | ' ||  LPAD('Date Generated: ', 76)
            || TO_CHAR(SYSDATE, 'DD-MM-YYYY HH24:MI:SS') || ' | ');
   DBMS_OUTPUT.PUT_LINE(' | ' || RPAD(' ', 95) || ' | ');
   DBMS_OUTPUT.PUT_LINE('|' || LPAD('-', 97, '-') || '|');
   DBMS_OUTPUT.PUT_LINE('|' || RPAD('Customer Name', 30) || '|' ||
```

```
                        RPAD('Total Sales (RM)', 25) || '|' || RPAD('Year',20)
|| '|'|| RPAD('Top Sales Pointer',19)|| '|');
   DBMS_OUTPUT.PUT_LINE('|' || LPAD('-', 97, '-') || '|');

   FOR rec IN (
      SELECT CustomerID, CustomerName, Year, TotalSales
      FROM custPurchase
      WHERE TotalSales > 110000
      ORDER BY Year, TotalSales DESC
   ) LOOP
      v_customer_id := rec.CustomerID;
      v_customer_name := rec.CustomerName;
      v_year := rec.Year;
      v_total_sales := rec.TotalSales;

      IF v_year != v_max_year THEN
         v_max_total_sales := 0;
         v_max_year := v_year;
      END IF;

      IF v_total_sales > v_max_total_sales THEN
         v_max_total_sales := v_total_sales;
      END IF;

      IF v_total_sales = v_max_total_sales THEN
         DBMS_OUTPUT.PUT_LINE('|' || RPAD(v_customer_name, 30) ||
                              '|' || LPAD(TO_CHAR(v_total_sales,
'999,999,999.00'), 25) ||
                              '|' || LPAD(v_year, 20) || '|*'||LPAD('
',18)||'|');
      ELSE
         DBMS_OUTPUT.PUT_LINE('|' || RPAD(v_customer_name, 30) ||
                              '|' || LPAD(TO_CHAR(v_total_sales,
'999,999,999.00'), 25) ||
                              '|' || LPAD(v_year, 20) || '|'||LPAD('
',19)||'|');
      END IF;
   END LOOP;

   DBMS_OUTPUT.PUT_LINE('|' || LPAD('-', 97, '-') || '|');
   DBMS_OUTPUT.PUT_LINE('|' || LPAD('End of Report', 57) ||LPAD(' ',40) ||
'|');
   DBMS_OUTPUT.PUT_LINE(LPAD('=', 99, '='));
END;
/

spool C:\Users\sherm\OneDrive\Desktop\QUERY3_rpt.txt

EXEC Over110k;

spool off
```

Customers Spending Over RM110k by Year

Date Generated: 30-04-2023 12:30:40

Customer Name	Total Sales (RM)	Year	Top Sales Pointer
WellCare Health Plans	122,287.10	2018	*
Colgate-Palmolive	120,024.00	2018	
FedEx	111,060.69	2018	
Charter Communications	120,057.87	2019	*
Computer Sciences	115,368.28	2019	
WestRock	110,374.93	2019	
Owens & Minor	137,130.80	2020	*
Oracle	129,781.71	2020	
National Oilwell Varco	129,232.18	2020	
US Foods Holding	119,190.53	2020	
Henry Schein	113,853.33	2020	
Illinois Tool Works	110,992.38	2020	
Sonic Automotive	152,230.56	2021	*
Cigna	129,753.14	2021	
Monsanto	122,390.00	2021	
Applied Materials	119,847.18	2021	
Danaher	117,816.05	2021	
Kinder Morgan	115,677.09	2021	
UnitedHealth Group	115,086.54	2021	
Performance Food Group	115,083.66	2021	
AbbVie	114,796.42	2021	
Alphabet	112,845.96	2021	
Bristol-Myers Squibb	110,552.46	2021	
Marsh & McLennan	110,234.35	2021	
National Oilwell Varco	110,112.39	2021	
Devon Energy	132,366.90	2022	*
United Technologies	125,145.72	2022	
Jabil Circuit	114,079.32	2022	

End of Report

The diagram above shows the customers who spend over RM110 thousands each year. The years included are 2018, 2019, 2020, 2021, 2022. The top sales pointer indicates the highest sales value for that particular year.

Conclusion, the top spend customer who spent over RM110 thousands for 2018 is WellCare Health Plans with a value of RM122,287.10. For 2019, it shows that Charter Communication is the highest spend with RM120,057.87. In 2020, Owens & Minor is the highest with the value of RM137,130.80. For 2021 and 2022, Sonic Automotive and Devon Energy are the highest spenders in this category with value of RM152,230.56 and RM132,366.90 respectively.

3.3 Sit Yie Sian

3.3.1 Report 1: Top sales state in 2021 and their city sales percentage growth for Q1 2021 and 2022

The objective of this report is to show the top-performing state (based on sales in 2021) to understand how the sales have changed between Q1 2021 and Q1 2022 for each city within that state. The query calculates the total sales, sales difference, and percentage growth for each city and provides a summary row with the aggregated values for the entire state.

This analysis is important for businesses because it helps them understand the performance of their sales in various cities within their top-performing state. By identifying the cities with the highest growth or decline, businesses can make informed decisions about where to focus their marketing efforts, allocate resources, or expand their presence. Additionally, understanding the overall state-level performance can help businesses assess the effectiveness of their regional strategies and identify opportunities for improvement.

SQL Query:

```
CREATE OR REPLACE VIEW sales_summary AS
  SELECT
    c.state,
    c.city,
    SUM(s.total_amount) AS total_sales,
    d.cal_year,
    d.cal_year_qtr
  FROM
    sales_fact s
    JOIN date_dim d ON s.date_key = d.date_key
    JOIN customer_dim c ON s.customer_key = c.customer_key
  WHERE
    d.cal_year IN (2021, 2022)
  GROUP BY
    c.state, c.city, d.cal_year, d.cal_year_qtr;

CREATE OR REPLACE VIEW state_sales AS
  SELECT
    state,
    SUM(total_sales) AS total_sales,
    RANK() OVER (ORDER BY SUM(total_sales) DESC) AS sales_rank
  FROM
    sales_summary
  WHERE
    cal_year = 2021
  GROUP BY
    state;

CREATE OR REPLACE VIEW city_sales AS
  SELECT
    state,
```

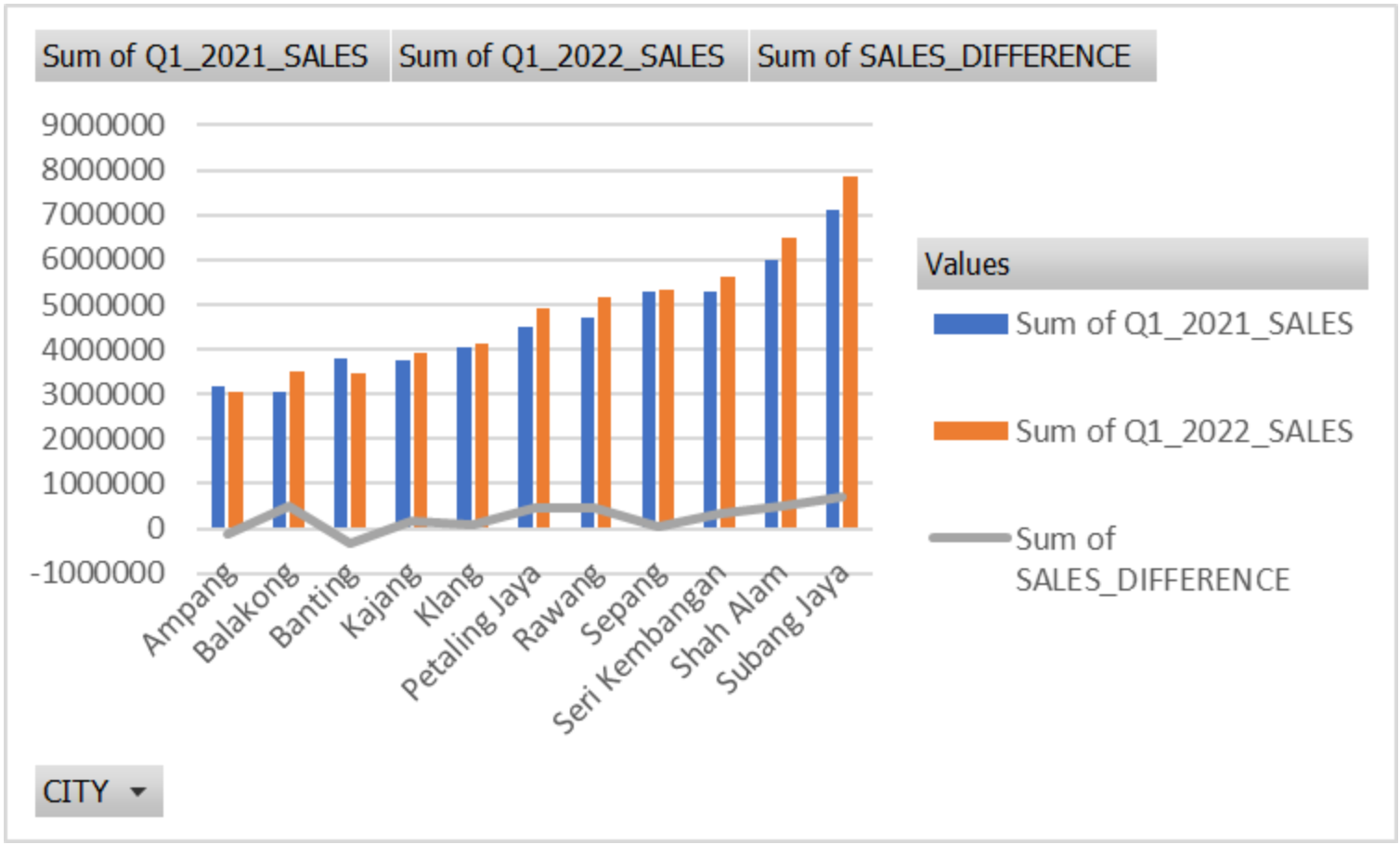
```
    city,
    SUM(CASE WHEN cal_year_qtr = '2021Q1' THEN total_sales ELSE 0 END) AS
q1_2021_sales,
    SUM(CASE WHEN cal_year_qtr = '2022Q1' THEN total_sales ELSE 0 END) AS
q1_2022_sales
  FROM
    sales_summary
  WHERE
    cal_year_qtr IN ('2021Q1', '2022Q1')
  GROUP BY
    state, city;

CREATE OR REPLACE VIEW top_state AS
  SELECT
    state
  FROM
    state_sales
  WHERE
    sales_rank = 1;

CREATE OR REPLACE VIEW city_percentage_growth AS
  SELECT
    cs.state,
    cs.city,
    cs.q1_2021_sales,
    cs.q1_2022_sales,
    cs.q1_2022_sales - cs.q1_2021_sales AS sales_difference,
    ((cs.q1_2022_sales - cs.q1_2021_sales) / cs.q1_2021_sales) * 100 AS
percentage_growth
  FROM
    city_sales cs
    JOIN top_state ts ON cs.state = ts.state;

col state               format a10 trunc
col city                format a16 trunc
col percentage_growth   format a17 trunc

set trimspool ON
spool d:\sys_query1.txt

set linesize 120
set pagesize 35

SELECT
  state,
  COALESCE(city, 'Total') AS city,
  SUM(q1_2021_sales) AS q1_2021_sales,
  SUM(q1_2022_sales) AS q1_2022_sales,
  SUM(sales_difference) AS sales_difference,
  ROUND(AVG(percentage_growth), 2) || '%' AS percentage_growth
FROM
  city_percentage_growth
GROUP BY ROLLUP (state, city)
HAVING
  NOT (city IS NULL AND state IS NULL)
ORDER BY
  city;
```

```
spool off
```

Output:

STATE	CITY	Q1_2021_SALES	Q1_2022_SALES	SALES_DIFFERENCE	PERCENTAGE_GROWTH
Selangor	Ampang	3175703.81	3069476.12	-106227.69	-3.35%
Selangor	Balakong	3041871.51	3522265.79	480394.28	15.79%
Selangor	Banting	3795957.91	3452299.99	-343657.92	-9.05%
Selangor	Kajang	3752745.59	3934132.31	181386.72	4.83%
Selangor	Klang	4045677.47	4146039.68	100362.21	2.48%
Selangor	Petaling Jaya	4490971.99	4929478.36	438506.37	9.76%
Selangor	Rawang	4702193.99	5165239.6	463045.61	9.85%
Selangor	Sepang	5280837.26	5336622.28	55785.02	1.06%
Selangor	Seri Kembangan	5293567.74	5610448.13	316880.39	5.99%
Selangor	Shah Alam	5982768.29	6493263.33	510495.04	8.53%
Selangor	Subang Jaya	7129078.25	7841747.68	712669.43	10%
Selangor	Total	50691373.8	53501013.3	2809639.46	5.08%

12 rows selected.


Sum of Q1_2021_SALES
Sum of Q1_2022_SALES
Sum of SALES_DIFFERENCE
9000000
8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
-1000000
Ampang
Balakong
Banting
Kajang
Klang
Petaling Jaya
Rawang
Sepang
Seri Kembangan
Shah Alam
Subang Jaya
Values
Sum of Q1_2021_SALES
Sum of Q1_2022_SALES
Sum of SALES_DIFFERENCE
CITY

3.3.2 Report 2: Top 10 employee sales in 2020 and compare their sales performance in 2021

The objective of this report is to show the sales performance of the top 10 employees in the year 2020 and compare their sales in 2021. The query calculates the total sales, sales difference, and percentage growth for each employee in the top 10, as well as a summary row with the aggregated values for all top 10 employees.

This analysis is important for businesses because it helps them understand the performance of their top sales employees and how their sales have changed over time. By identifying the employees with the highest growth or decline, businesses can make informed decisions about recognizing top performers, providing additional support or training to employees who are struggling, or even adjusting their sales strategies. Additionally, understanding the overall performance of the top 10 employees can help businesses assess the effectiveness of their sales team and identify opportunities for improvement.

```
SQL Query:
CREATE OR REPLACE VIEW sales_summary AS
  SELECT
    e.employee_id,
    d.cal_year AS year,
    sum(s.total_amount) AS sales
  FROM
    sales_fact s
    JOIN date_dim d ON s.date_key = d.date_key
    JOIN employee_dim e ON s.employee_key = e.employee_key
  WHERE
    d.cal_year IN (2020,2021)
  GROUP BY
    e.employee_id, d.cal_year;

CREATE OR REPLACE VIEW employee_sales_yearly AS
  SELECT
    employee_id,
    year,
    sales,
    LAG(sales) OVER (PARTITION BY employee_id ORDER BY year) AS prev_year_sales
  FROM
    sales_summary;

CREATE OR REPLACE VIEW top_10_2020 AS
  SELECT
    employee_id,
    year,
    sales,
    prev_year_sales
  FROM (
    SELECT
```

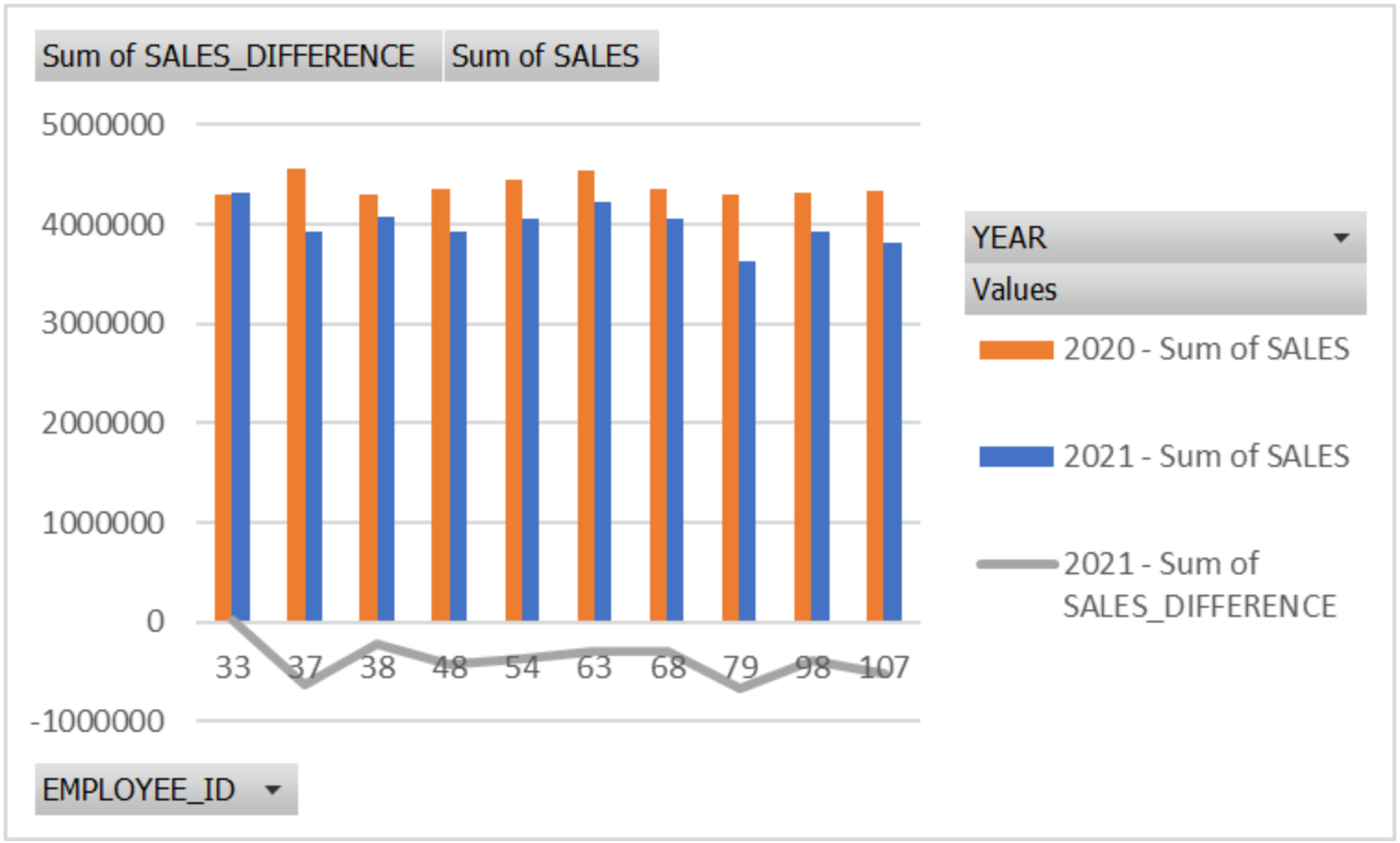
```
      employee_id,
      year,
      sales,
      prev_year_sales,
      ROW_NUMBER() OVER (ORDER BY sales DESC) AS row_num
    FROM
      employee_sales_yearly
    WHERE
      year = 2020
  ) subquery
  WHERE row_num <= 10;

CREATE OR REPLACE VIEW final_top_10 AS
  SELECT
    t.employee_id,
    esy.year,
    esy.sales,
    esy.prev_year_sales
  FROM
    employee_sales_yearly esy
    JOIN top_10_2020 t ON esy.employee_id = t.employee_id
  WHERE
    esy.year IN (2020, 2021);

CREATE OR REPLACE VIEW top_10_2020_and_2021 AS
  SELECT
    employee_id,
    year,
    sales,
    prev_year_sales,
    (CASE WHEN year = 2020 THEN sales END) AS sales_2020
  FROM
    final_top_10;

CREATE OR REPLACE VIEW ordered_top_10_2020_and_2021 AS
  SELECT
    employee_id,
    year,
    sales,
    prev_year_sales,
    sales - prev_year_sales AS sales_difference,
    ROUND(((sales - prev_year_sales) / prev_year_sales) * 100, 2) AS
percentage_growth
  FROM
    top_10_2020_and_2021
  ORDER BY
    COALESCE(sales_2020, LAG(sales_2020) OVER (PARTITION BY employee_id ORDER
BY year)) DESC,
    year;

CREATE OR REPLACE VIEW total_percentage_growth AS
  SELECT
    year,
    SUM(sales) AS total_sales,
    LAG(SUM(sales)) OVER (ORDER BY year) AS prev_year_total_sales
  FROM
    ordered_top_10_2020_and_2021
```

```
  GROUP BY
    year;

col employee_id format a7 trunc

spool d:\sys_query2.txt

set linesize 120
set pagesize 35

SELECT
  TO_CHAR(employee_id) AS employee_id,
  year,
  sales,
  prev_year_sales,
  sales_difference,
  TO_CHAR(ROUND(percentage_growth, 2)) AS percentage_growth
FROM
  ordered_top_10_2020_and_2021
UNION ALL
SELECT
  'Total' AS employee_id,
  year,
  total_sales,
  prev_year_total_sales,
  total_sales - prev_year_total_sales AS sales_difference,
  TO_CHAR(ROUND(((total_sales - prev_year_total_sales) / prev_year_total_sales)
* 100, 2)) AS percentage_growth
FROM
  total_percentage_growth
ORDER BY
  employee_id,
  year;

spool off
```

Output:

EMPLOYEE_ID	YEAR	SALES	PREV_YEAR_SALES	SALES_DIFFERENCE	PERCENTAGE_GROWTH
107	2020	4343692.56			
107	2021	3823576.21	4343692.56	-520116.35	-11.97
33	2020	4298159.13			
33	2021	4313687.9	4298159.13	15528.77	.36
37	2020	4564394.51			
37	2021	3925025.6	4564394.51	-639368.91	-14.01
38	2020	4306545.5			
38	2021	4081831.98	4306545.5	-224713.52	-5.22
48	2020	4359468.11			
48	2021	3929229.76	4359468.11	-430238.35	-9.87
54	2020	4443064.21			
54	2021	4066122.67	4443064.21	-376941.54	-8.48
63	2020	4535386.97			
63	2021	4231008.59	4535386.97	-304378.38	-6.71
68	2020	4357910			
68	2021	4056384	4357910	-301526	-6.92
79	2020	4300530.22			
79	2021	3622770.01	4300530.22	-677760.21	-15.76
98	2020	4314496.79			
98	2021	3925239.29	4314496.79	-389257.5	-9.02
Total	2020	43823648			
Total	2021	39974876	43823648	-3848772	-8.78

22 rows selected.


Sum of SALES_DIFFERENCE
Sum of SALES
5000000
4000000
3000000
2000000
1000000
0
-1000000
33
37
38
48
54
63
68
79
98
107
EMPLOYEE_ID
YEAR
Values
2020 - Sum of SALES
2021 - Sum of SALES
2021 - Sum of SALES_DIFFERENCE

3.3.3 Report 3 : Top 10 product overall sales and comparing the percentage sales in weekend and weekdays

The objective of this report is to show the sales performance of the top 10 products by overall sales, comparing the sales on weekends and weekdays. The query calculates the sales, as well as the percentage of sales that occur on weekends and weekdays for each of the top 10 products. Additionally, a summary row with the aggregated values for all top 10 products is included.

This analysis is important for businesses because it helps them understand the performance of their top-selling products and how sales are distributed between weekends and weekdays. By identifying the products with higher sales on weekends or weekdays, businesses can make informed decisions about promotional strategies, inventory management, or adjusting their marketing efforts. Additionally, understanding the overall sales distribution between weekends and weekdays for the top 10 products can help businesses identify trends or patterns in customer purchasing behavior and make data-driven decisions to maximize sales and profitability.

SQL Query:

```
CREATE OR REPLACE VIEW product_sales_summary AS
  SELECT
    p.product_id,
    p.product_name,
    SUM(s.total_amount) AS total_sales
  FROM
    sales_fact s
    JOIN product_dim p ON s.product_key = p.product_key
  GROUP BY
    p.product_id, p.product_name;

CREATE OR REPLACE VIEW top_10_product_sales AS
  SELECT
    product_id,
    product_name,
    total_sales,
    ROW_NUMBER() OVER (ORDER BY total_sales DESC) AS row_num
  FROM
    product_sales_summary
  WHERE
    total_sales IS NOT NULL AND ROWNUM <= 10
  ORDER BY
    total_sales DESC;

CREATE OR REPLACE VIEW product_sales_comparison AS
  SELECT
    t.product_id,
    t.product_name,
    t.total_sales AS overall_sales,
    wp.weekend_sales,
    TO_CHAR(ROUND(wp.weekend_sales / t.total_sales * 100, 2)) || '%' AS
pct_weekend_sales,
```

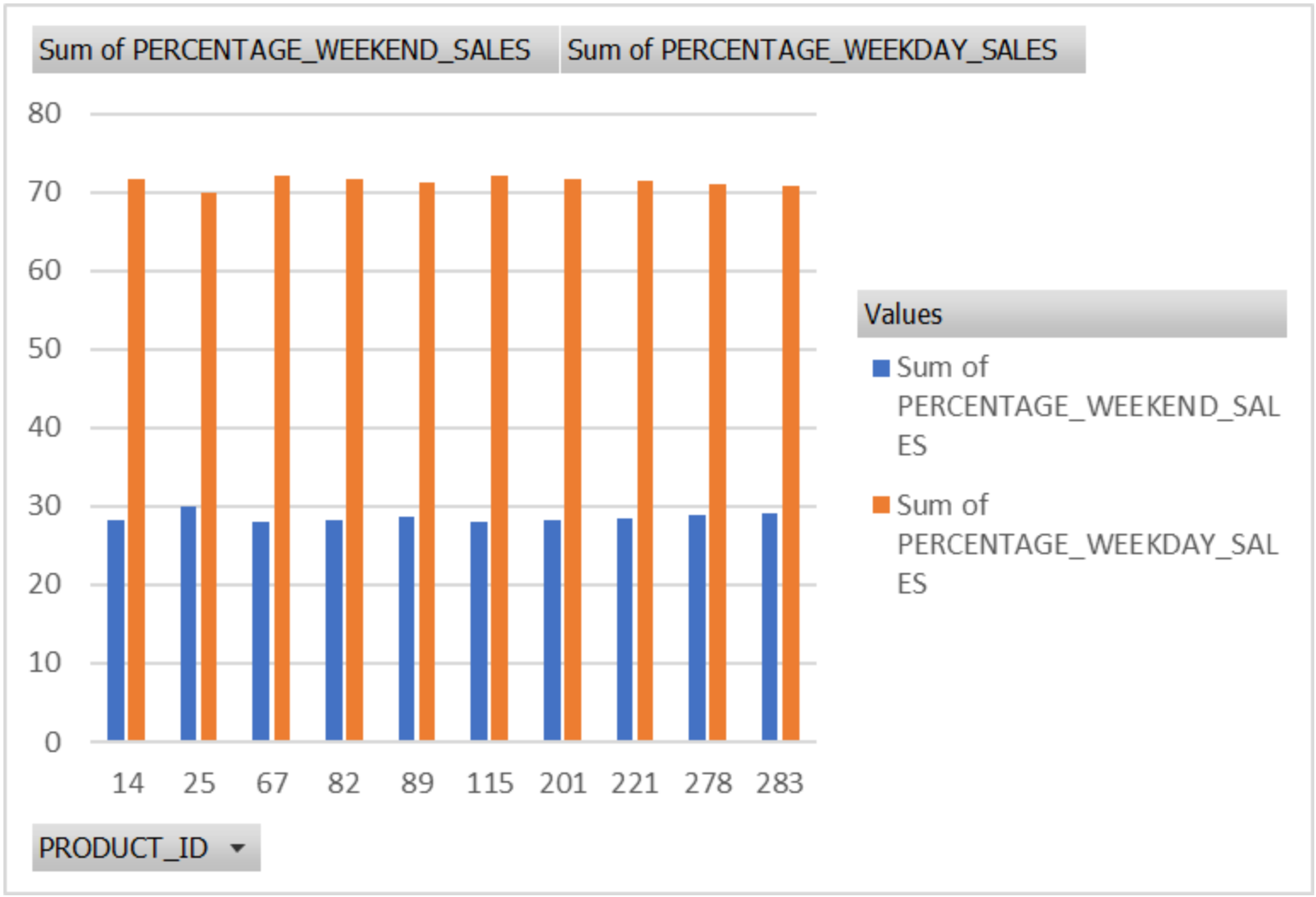
```
    wd.weekday_sales,
    TO_CHAR(ROUND(wd.weekday_sales / t.total_sales * 100, 2)) || '%' AS
pct_weekday_sales
  FROM
    top_10_product_sales t
    LEFT JOIN (
      SELECT
        p.product_id,
        SUM(s.total_amount) AS weekend_sales
      FROM
        sales_fact s
        JOIN date_dim d ON s.date_key = d.date_key
        JOIN product_dim p ON s.product_key = p.product_key
      WHERE
        d.weekday_ind = 'N'
      GROUP BY
        p.product_id
    ) wp ON t.product_id = wp.product_id
    LEFT JOIN (
      SELECT
        p.product_id,
        SUM(s.total_amount) AS weekday_sales
      FROM
        sales_fact s
        JOIN date_dim d ON s.date_key = d.date_key
        JOIN product_dim p ON s.product_key = p.product_key
      WHERE
        d.weekday_ind = 'Y'
      GROUP BY
        p.product_id
    ) wd ON t.product_id = wd.product_id;

col product_name      format a25 trunc
col pct_weekend_sales format a17 trunc
col pct_weekday_sales format a17 trunc

set linesize 120
set pagesize 35

spool d:\sys_query3.txt

SELECT
  product_id,
  product_name,
  overall_sales,
  weekend_sales,
  pct_weekend_sales,
  weekday_sales,
  pct_weekday_sales
FROM
  product_sales_comparison
UNION ALL
SELECT
  NULL AS product_id,
  'Total' AS product_name,
  SUM(overall_sales) AS overall_sales,
  SUM(weekend_sales) AS weekend_sales,
```

```
  TO_CHAR(ROUND(SUM(weekend_sales) / SUM(overall_sales) * 100, 2)) || '%' AS
pct_weekend_sales,
  SUM(weekday_sales) AS weekday_sales,
  TO_CHAR(ROUND(SUM(weekday_sales) / SUM(overall_sales) * 100, 2)) || '%' AS
pct_weekday_sales
FROM
  product_sales_comparison;

spool off
```

Output:

PRODUCT_ID	PRODUCT_NAME	OVERALL_SALES	WEEKEND_SALES	PCT_WEEKEND_SALES	WEEKDAY_SALES	PCT_WEEKDAY_SALES
25	Samsung MZ-75E250B/AM	868825.92	260417.04	29.97%	608408.88	70.03%
14	G.Skill Ripjaws V Series	6104508.41	1726520.51	28.28%	4377987.9	71.72%
283	G.Skill Trident Z	5921263.19	1726686.31	29.16%	4194576.88	70.84%
201	Kingston	5535798.5	1563825.5	28.25%	3971973	71.75%
221	Zotac ZT-P10810C-10P	6768470.94	1932654.57	28.55%	4835816.37	71.45%
115	Corsair Vengeance LPX	5945715.06	1661076.27	27.94%	4284638.79	72.06%
278	G.Skill Ripjaws V Series	5811052.29	1677347.26	28.86%	4133705.03	71.14%
89	PNY VCGGTX780T3XPB-OC	6576662.31	1888474.82	28.71%	4688187.49	71.29%
67	Kingston HyperX Beast	7364405.65	2059237.3	27.96%	5305168.35	72.04%
82	Intel Core i7-6950X	12714567.5	3589236.77	28.23%	9125330.76	71.77%
	Total	63611269.8	18085476.4	28.43%	45525793.5	71.57%

11 rows selected.


Sum of PERCENTAGE_WEEKEND_SALES
Sum of PERCENTAGE_WEEKDAY_SALES
80
70
60
50
40
30
20
10
0
14
25
67
82
89
115
201
221
278
283
PRODUCT_ID
Values
Sum of PERCENTAGE_WEEKEND_SALES
Sum of PERCENTAGE_WEEKDAY_SALES

3.4 Tan Jacqueline

3.4.1 Report I: Monthly best city performance report

This report indicates the monthly best performance city in a year. This report are able to compare the performance among the cities in each state. The total sales of the year have split into 12 different months to show the difference between them. The organization might use this report to distribute the rewards or commission. By viewing this report,the company is able to see which city has performed the best and how many percent of their contribution in the company's total sales in that particular year.

SQL Query :

```
-- Monthly Sales Performance by Office
set serveroutput on
set linesize 200
set pagesize 50

create or replace view Monthly_STATE_Sales_vw as
      select D.CAL_YEAR AS Year, D.MONTH_NUMBER AS MonthNo,
      C.STATE AS State, C.CITY AS City,SUM(Total_Amount) as Total
      from SALES_FACT SF
            join DATE_DIM D
                  on SF.DATE_KEY = D.DATE_KEY
            join CUSTOMER_DIM C
                  on SF.CUSTOMER_KEY = C.CUSTOMER_KEY
            group by D.CAL_YEAR, D.MONTH_NUMBER, C.STATE, C.CITY
            order by D.CAL_YEAR, D.MONTH_NUMBER,
            SUM(Total_Amount)DESC;



create or replace view Month_vw as
      SELECT CAL_YEAR AS Year, MONTH_NUMBER AS MonthNo,
      MONTH_NAME AS Month
      FROM DATE_DIM
      GROUP By CAL_YEAR, MONTH_NUMBER, MONTH_NAME
      ORDER BY CAL_YEAR, MONTH_NUMBER;



create or replace view Total_sales_per_month_vw as
      select D.CAL_YEAR AS Year, D.MONTH_NUMBER AS MonthNo,
      SUM(Total_Amount) as Total
      from SALES_FACT SF
            join DATE_DIM D
                  on SF.DATE_KEY = D.DATE_KEY
      group by D.CAL_YEAR, D.MONTH_NUMBER
      order by D.CAL_YEAR, D.MONTH_NUMBER;
```

```
create or replace view top_monthly_office_sales_vw as
      select D.CAL_YEAR AS Year, D.MONTH_NAME AS Month,
      D.MONTH_NUMBER AS MonthNo, C.STATE AS State, C.CITY AS
      City, SUM(Total_Amount) as Total
      from SALES_FACT SF
            join DATE_DIM D
                  on SF.DATE_KEY = D.DATE_KEY
            join CUSTOMER_DIM C
                  on SF.CUSTOMER_KEY = C.CUSTOMER_KEY
      group by D.CAL_YEAR, D.MONTH_NAME, D.MONTH_NUMBER, C.STATE,
      C.CITY
      order by D.CAL_YEAR, SUM(Total_Amount) DESC;



create or replace view Total_sales_per_year_vw as
      select D.CAL_YEAR AS Year, SUM(Total_Amount) as Total from
      SALES_FACT SF
            join DATE_DIM D
                  on SF.DATE_KEY = D.DATE_KEY
      group by D.CAL_YEAR
      order by D.CAL_YEAR;




CREATE OR REPLACE PROCEDURE MONTHLY_CITY_PERFORMANCE (in_year_no in number)IS

      v_monthNo         DATE_DIM.MONTH_NUMBER%TYPE;
      v_monthName       DATE_DIM.MONTH_NAME%TYPE;
      v_state           CUSTOMER_DIM.STATE%TYPE;
      v_city                  CUSTOMER_DIM.CITY%TYPE;
      v_total_sales_city   Monthly_City_Sales_vw.Total%TYPE;
      v_total_sales           Total_sales_per_month_vw.Total%TYPE;
      v_cont NUMBER(9,2):= 0;
      v_best_month               DATE_DIM.MONTH_NAME%TYPE;
      v_best_state               CUSTOMER_DIM.STATE%TYPE;
      v_best_city                CUSTOMER_DIM.CITY%TYPE;
      v_best_total_sales_city   Monthly_city_Sales_vw.Total%TYPE;
      v_best_cont NUMBER(9,2):= 0;
      v_best_monthNo             DATE_DIM.MONTH_NUMBER%TYPE;

      cursor Month_cur is
            select MonthNo, Month
            from Month_vw
            where Year = in_year_no
            Order by MonthNo ASC;

      cursor city_sales_cur is
            select MonthNo, City,State,  Total
            from Monthly_city_Sales_vw
            where Year = in_year_no
            AND MonthNo = v_monthNo
            AND rowNum <=1;
```

```
	cursor total_sales_per_month_cur IS
		select MonthNo, Total
		from Total_sales_per_month_vw
		where Year = in_year_no
		AND MonthNo = v_monthNo;

	cursor Total_sales_cur is
		select Total
		from Total_sales_per_year_vw
		where Year = in_year_no;

	cursor Total_sales_month_cur is
		select MonthNo, Month, City,State, Total
		from top_monthly_city_sales_vw
		where Year = in_year_no
		AND rowNum <=1;
Begin
	DBMS_OUTPUT.PUT_LINE(LPAD ('=',105,'='));
	DBMS_OUTPUT.PUT_LINE(' | '||LPAD('Computer Product
	Supplier',65,' ')|| LPAD(' | ',39));
	DBMS_OUTPUT.PUT_LINE(' | '||LPAD('Monthly Best Performance
	City in',67,' ') || ' ' || in_year_no||LPAD(' | ',32));
	DBMS_OUTPUT.PUT_LINE(' | '|| RPAD(' ',101) ||' | ');
	DBMS_OUTPUT.PUT_LINE(' | '||LPAD('Report Generated: ' ,83,
	' ')|| TO_CHAR(SYSDATE, 'DD-MM-YYYY HH24:MI:SS') || '|');
	DBMS_OUTPUT.PUT_LINE(' | '|| RPAD(' ',101) ||' | ');
	DBMS_OUTPUT.PUT_LINE(LPAD ('-',105,'-'));
	DBMS_OUTPUT.PUT_LINE(' |   Month   |          State
	|        City      |   Total Sales    |    Monthly Total
	Sales    |');
	DBMS_OUTPUT.PUT_LINE(LPAD ('-',105,'-'));
	OPEN Month_cur;
	FETCH Month_cur INTO v_monthNo, v_monthName;
	WHILE (Month_cur%FOUND)
	LOOP
		OPEN city_sales_cur;
		FETCH city_sales_cur INTO v_monthNo, v_city, v_state,
		V_total_sales_city;
	WHILE (city_sales_cur%FOUND)
		LOOP
			OPEN total_sales_per_month_cur;
			FETCH total_sales_per_month_cur INTO v_monthNo,
			V_total_sales;
			v_cont := v_total_sales_city/v_total_sales * 100;
			DBMS_OUTPUT.PUT_LINE(' | '||RPAD(v_monthName ,9,
			' ')|| ' |   ' ||RPAD(v_state ,21, ' ')||' |
			'||RPAD(v_city ,16, ' ')||' | '||
			RPAD(to_char(v_total_sales_city,
			'$99,999,999.00'),17)|| '|
			'||RPAD(to_char(v_total_sales,
			'$99,999,999.00'),17)||'('||
			RPAD(CONCAT(to_char(v_cont, 'fm990D00'), '%'),
			6)||')'||' |');

		CLOSE total_sales_per_month_cur;
			IF (v_total_sales_city > v_best_total_sales_city)
```

```
                THEN
                v_best_total_sales_city := v_total_sales_city;
                v_best_monthNo:= v_monthNo;
                END IF;
          FETCH city_sales_cur INTO v_monthNo, v_city, v_state,
          V_total_sales_city;
                v_cont := 0;
          END LOOP;
          CLOSE city_sales_cur;
          FETCH Month_cur INTO v_monthNo, v_monthName;
          END LOOP;
          CLOSE Month_cur;
    DBMS_OUTPUT.PUT_LINE(LPAD ('-',105,'-'));
    open Total_sales_cur;
    Fetch Total_sales_cur INTO v_total_sales;
    CLOSE Total_sales_cur;

    DBMS_OUTPUT.PUT_LINE(' | '||LPAD('Total Sales in (' ,77, '
    ')||in_year_no||'): '||RPAD(to_char(v_total_sales,
    '$999,999,999.00'),17)|| ' |');
    DBMS_OUTPUT.PUT_LINE(' | '|| RPAD(' ',101) ||' | ');
    DBMS_OUTPUT.PUT_LINE(' |-'||LPAD(' Summary ',53,'-')||
    LPAD('-|',50, '-'));
    DBMS_OUTPUT.PUT_LINE(' | '||' As compared to monthly sales
    in '||in_year_no|| LPAD('|',50, '')||LPAD('|',66, ' '));
    OPEN Total_sales_month_cur;
    FETCH Total_sales_month_cur INTO v_monthNo, v_monthName,
    v_city, v_state, v_total_sales_city;
          WHILE (Total_sales_month_cur%FOUND)
          LOOP
          OPEN total_sales_per_month_cur;
          FETCH total_sales_per_month_cur INTO v_monthNo,
          V_total_sales;
          v_cont := v_total_sales_city/v_total_sales * 100;
          DBMS_OUTPUT.PUT_LINE(' | '||' The Month with the
          highest sales is '||v_monthName||' which having
          (RM)'||RPAD(to_char(v_total_sales_city,
          '$9,999,999.00'),14)||' ('||RPAD
          (CONCAT(to_char(v_cont, 'fm990D00'), '%'), 6)||')'||
          LPAD('|',50, '')||LPAD(' |',16, ' '));

          DBMS_OUTPUT.PUT_LINE(' | '||' The best city with the
          highest sales is '||v_city||' which located in
          '||v_state||LPAD(' |',13, ' '));
          CLOSE total_sales_per_month_cur;
          FETCH Total_sales_month_cur INTO v_monthNo,
          v_monthName, v_city, v_state, v_total_sales_city;
          v_cont := 0;
          END LOOP;
          CLOSE Total_sales_month_cur;

    DBMS_OUTPUT.PUT_LINE(' | '|| RPAD(' ',101) ||' | ');
    DBMS_OUTPUT.PUT_LINE(' | '||LPAD(' <END OF REPORT>',65,'
    ')|| LPAD(' | ',39));
    DBMS_OUTPUT.PUT_LINE(LPAD ('=',105,'='));
    DBMS_OUTPUT.PUT_LINE(chr(008));
END;
```

```
/
exec MONTHLY_CITY_PERFORMANCE(2021);
```

```
SQL> exec MONTHLY_CITY_PERFORMANCE(2021);
```

Computer Product Supplier
Monthly Best Performance City in 2021

Report Generated: 30-04-2023 11:23:43

Month	State	City	Total Sales	Monthly Total Sales
January	Wilayah Persekutuan	Kuala Lumpur	$3,362,212.22	$34,775,620.74 (9.67%)
February	Wilayah Persekutuan	Kuala Lumpur	$3,975,530.31	$37,509,848.48 (10.60%)
March	Wilayah Persekutuan	Kuala Lumpur	$3,955,684.82	$36,683,707.58 (10.78%)
April	Wilayah Persekutuan	Kuala Lumpur	$3,796,777.33	$35,109,424.35 (10.81%)
May	Wilayah Persekutuan	Kuala Lumpur	$4,053,614.41	$40,375,056.41 (10.04%)
June	Wilayah Persekutuan	Kuala Lumpur	$3,913,484.28	$33,525,059.68 (11.67%)
July	Wilayah Persekutuan	Kuala Lumpur	$3,315,698.51	$30,856,708.95 (10.75%)
August	Wilayah Persekutuan	Kuala Lumpur	$3,972,907.68	$36,629,506.60 (10.85%)
September	Wilayah Persekutuan	Kuala Lumpur	$4,110,074.95	$37,129,722.79 (11.07%)
October	Wilayah Persekutuan	Kuala Lumpur	$3,663,616.40	$36,399,361.06 (10.07%)
November	Wilayah Persekutuan	Kuala Lumpur	$3,406,018.31	$30,575,697.80 (11.14%)
December	Wilayah Persekutuan	Kuala Lumpur	$3,493,690.80	$33,864,503.17 (10.32%)

Total Sales in (2021): $423,434,217.61

Summary

As compared to monthly sales in 2021
The Month with the highest sales is September which having (RM) $4,110,074.95 (11.07%)
The best city with the highest sales is Kuala Lumpur which located in Wilayah Persekutuan

<END OF REPORT>

In this report, the year will be the input and show the date and time of the report generated. Next it will display the month of the year and the state that the city is located in and calculate the total sales and how many percent it contributes. In 2021, the city with the best performance will be Kuala Lumpur, which is located in Wilayah Persekutuan in September.

3.4.2 Report 2:Top 10 employee of the year

This report indicates the top 10 employees that have the highest performance based on the total sales that the staff has handled. The details of the employee such as Employee ID, Name which includes the first name and last name, the job title of the employee and the total sales of the employee has been made will be shown. WIth this report, the company is able to view which 10 employees have performed the best. Lastly, a simple summary will be displayed. It will display the best performance staff and is total sales plus its contribution based on the total sales.

SQL Query:

```
-- Top 10 employees with highest sales
set serveroutput on
set linesize 200
set pagesize 50

create or replace view Emp_Sales_vw as
select D.CAL_YEAR AS Year, C.EMPLOYEE_ID AS EmployeeID, SUM(Total_Amount) as
Total
      from SALES_FACT SF
            join DATE_DIM D
                  on SF.DATE_KEY = D.DATE_KEY
            join EMPLOYEE_DIM C
                  on SF.EMPLOYEE_KEY = C.EMPLOYEE_KEY
            join CUSTOMER_DIM D
                  on SF.CUSTOMER_KEY = D.CUSTOMER_KEY
            join ORDERS E
                  on D.CUSTOMER_ID = E.CUSTOMER_ID
            group by D.CAL_YEAR, C.EMPLOYEE_ID
            order by D.CAL_YEAR, SUM(Total_Amount) DESC;

create or replace view Emp_Details_vw as
select C.EMPLOYEE_ID AS EmployeeID,C.Last_name AS LastName, C.First_name AS
FirstName, Job_title
      from SALES_FACT SF
            join DATE_DIM D
                  on SF.DATE_KEY = D.DATE_KEY
            join EMPLOYEE_DIM C
                  on SF.EMPLOYEE_KEY = C.EMPLOYEE_KEY
      group by D.CAL_YEAR, C.EMPLOYEE_ID, C.First_name, C.Last_name, Job_title
      order by D.CAL_YEAR, C.EMPLOYEE_ID, SUM(Total_Amount) DESC;



CREATE OR REPLACE PROCEDURE TOP_10_STAFF_PERFORMANCE (in_year_no in number)IS
      v_employeeId EMPLOYEE_DIM.EMPLOYEE_ID%TYPE;
      v_total Emp_Sales_vw.Total%TYPE;
      v_lastName Emp_Details_vw.LASTNAME%TYPE;
      v_firstName Emp_Details_vw.FIRSTNAME%TYPE;
      v_avg_sales NUMBER(9,2):= 0;
```

```
        v_total_sales Total_sales_per_year_vw.Total%TYPE;
        v_position Emp_Details_vw.Job_title%TYPE;
        v_employee_cnt NUMBER(3):= 0;
        v_best_lastName Emp_Details_vw.LASTNAME%TYPE;
        v_best_firstName Emp_Details_vw.FIRSTNAME%TYPE;
        v_cont NUMBER(9,2):= 0;

        cursor Emp_Sales_cur is
              select EmployeeID, Total
              From Emp_Sales_vw
              where year = in_year_no AND rownum <= 10
              order by Total DESC;

        cursor Emp_details_cur is
              select *
              From Emp_Details_vw
              where EmployeeID = v_employeeId;

Begin
        DBMS_OUTPUT.PUT_LINE(LPAD ('=',91,'='));
        DBMS_OUTPUT.PUT_LINE(' | '||LPAD('Computer Product Supplier',59,' ')||
        LPAD(' | ',31));
        DBMS_OUTPUT.PUT_LINE(' | '||LPAD('Top 10 Staff Performance',56,' ') || '
        ' || in_year_no||LPAD(' | ',29));
        DBMS_OUTPUT.PUT_LINE(' | '|| RPAD(' ',87) ||' | ');
        DBMS_OUTPUT.PUT_LINE(' | '||LPAD('Report Generated: ' ,68, ' ')||
        TO_CHAR(SYSDATE, 'DD-MM-YYYY HH24:MI:SS')||' | ');
        DBMS_OUTPUT.PUT_LINE(' | '|| RPAD(' ',87) ||' | ');
        DBMS_OUTPUT.PUT_LINE(LPAD ('-',91,'-'));
        DBMS_OUTPUT.PUT_LINE(' |  ID   |      Full Name    |
        Position                |      Total (RM)      |');
        DBMS_OUTPUT.PUT_LINE(LPAD ('-',91,'-'));
        OPEN Emp_Sales_cur;
        FETCH Emp_Sales_cur into v_employeeId, v_total;
        WHILE (Emp_Sales_cur%FOUND) LOOP
              OPEN Emp_details_cur;
              FETCH Emp_details_cur into v_employeeId,v_lastName, v_firstName,
              V_position;
              DBMS_OUTPUT.PUT_LINE(' | '||RPAD(v_employeeId ,6, ' ')||'| '
              ||RPAD(CONCAT(CONCAT(v_firstName, ' '), v_lastName), 18)||'|
              '||RPAD(v_position ,36, ' ')||' | '||RPAD(to_char(v_total,
              '$99,999,999.00'),21)|| '|');
              CLOSE Emp_details_cur;

              FETCH Emp_Sales_cur into v_employeeId, v_total;
        END LOOP;
        CLOSE Emp_Sales_cur;
        DBMS_OUTPUT.PUT_LINE(LPAD ('-',91,'-'));
        DBMS_OUTPUT.PUT_LINE(' | '|| RPAD(' ',87) ||' | ');
        DBMS_OUTPUT.PUT_LINE(' |-'||LPAD(' Summary ',47,'-')|| LPAD('-|',42,
        '-'));
        DBMS_OUTPUT.PUT_LINE(' | '||' As compared to overall staff performance
        in '||in_year_no|| LPAD('|',51, '')||LPAD(' |',40, ' '));
        OPEN Emp_Sales_cur;
        FETCH Emp_Sales_cur into v_employeeId, v_total;
              OPEN Emp_details_cur;
              FETCH Emp_details_cur into v_employeeId,
```

```
            v_firstName,v_lastName,v_position;
            DBMS_OUTPUT.PUT_LINE(' | '||' The best performance staff is
            '||RPAD(CONCAT(CONCAT(v_firstName, ' '), v_lastName),20)||
            LPAD('|',38, ' '));
            DBMS_OUTPUT.PUT_LINE(' | '||' The Sales is'||RPAD(to_char(v_total,
            '$99,999,999.00'),16)|| LPAD('|',60, ' '));
      CLOSE Emp_details_cur;
      CLOSE Emp_Sales_cur;

      DBMS_OUTPUT.PUT_LINE(' | '|| RPAD(' ',87) ||' | ');
      DBMS_OUTPUT.PUT_LINE(' | '||LPAD(' <END OF REPORT> ',51,' ')|| LPAD(' |
      ',39));
      DBMS_OUTPUT.PUT_LINE(LPAD ('=',91,'='));
      DBMS_OUTPUT.PUT_LINE(chr(008));
END;
/

exec TOP_10_STAFF_PERFORMANCE(2022);
```

```
SQL> exec TOP_10_STAFF_PERFORMANCE(2022);
```

```
==========================================================================================
|                                  Computer Product Supplier                             |
|                                Top 10 Staff Performance 2022                           |
|                                                                                        |
|                                              Report Generated: 30-04-2023 11:43:04 |
|                                                                                        |
------------------------------------------------------------------------------------------
|  ID   |      Full Name    |              Position               |      Total (RM)      |
------------------------------------------------------------------------------------------
| 69    | Evelyn Tucker     | Sales Representative                |  $80,030,792.19      |
| 70    | Eva Porter        | Sales Representative                |  $78,086,717.87      |
| 82    | Willow Reyes      | Shipping Clerk                      |  $76,068,176.56      |
| 96    | Hannah Knight     | Shipping Clerk                      |  $75,862,627.71      |
| 88    | Ellie Robertson   | Shipping Clerk                      |  $75,672,249.48      |
| 103   | Amelie Hudson     | Marketing Representative            |  $75,093,075.75      |
| 46    | Ava Sullivan      | Sales Manager                       |  $75,080,268.39      |
| 15    | Rory Kelly        | Purchasing Manager                  |  $74,989,514.68      |
| 107   | Summer Payne      | Public Accountant                   |  $74,780,809.70      |
| 37    | Ibrahim Alexander | Stock Clerk                         |  $74,625,537.90      |
```

```
-------------------------------------------------------------------------------------------
|                                                                                         |
|---------------------------------------- Summary ----------------------------------------|
|  As compared to overall staff performance in 2022                                       |
|  The best performance staff is Tucker Evelyn                                            |
|  The Sales is $80,030,792.19                                                            |
|                                                                                         |
|                                    <END OF REPORT>                                      |
===========================================================================================
```

We have placed the year as an input, as the output above indicates the result printed. The report will show the Report Generated Date and Time. Next, it will display the Employee ID, employee name, job title, total sales that have been handled by the employee. In conclusion, we can see under the summary session that as compared to the overall staff performance in 2021, the best performing staff is Tucker Evelyn. The sales is $80,030,792.19.

3.4.3 Report 3: Annual Sales of last three years and calculate the growth percentage

This report provides insights into the sales performance of various states of the company for the past three years, with the current year as the reference point. Its purpose is to aid long-term planning and strategy development. The ranking of states in the report is based on the total sales they have generated in the past three years, with the best-performing state appearing at the top and the worst-performing at the bottom. The report also shows the Year-On-Year (YoY) growth of each state over the two previous years (excluding the first year, which lacks a reference point for comparison). By analyzing this data, the company can determine which states are performing the best and which offices are experiencing the fastest growth in terms of YoY. The YoY calculation uses the formula below:

$$\text{Year-Over-Year Growth} = \left(\frac{\text{Current Period Amount} - \text{Previous Period Amount}}{\text{Previous Period Amount}} \right) \times 100$$

The report also presents the total sales for each year and the cumulative sales over the three-year period in a clear manner. The summary section of the report provides a concise overview of the sales performance and growth for each year. Growth is not calculated for the first year, as there is no previous data available for comparison. For the second and third years, the report identifies the state with the highest sales, the fastest-growing state, and the slowest-growing state based on YoY sales. This information can be used by the company to make informed decisions, such as investing more resources in the best-performing state and conducting descriptive analytics to identify the reasons why the worst-performing states are underperforming and why some states are growing slower than others.

SQL Query:

```
set serveroutput on
set linesize 200
set pagesize 50

CREATE OR REPLACE VIEW State_YoY_Vw AS
      SELECT C.STATE,D.CAL_YEAR Year,
             SUM(Total_Amount) as YearTotal,
             LAG(D.CAL_YEAR) OVER (ORDER BY C.State) PreviousYear,
             LAG(SUM(Total_Amount)) OVER (ORDER BY C.State)PastYearTotal,
             ROUND((SUM(Total_Amount)-LAG(SUM(Total_Amount))
             OVER (ORDER BY C.State))/
             LAG( SUM(Total_Amount))
             OVER (ORDER BY C.State) * 100, 2) as YoY
      FROM SALES_FACT SF
```

```
      JOIN DATE_DIM D
             ON SF.DATE_KEY = D.DATE_KEY
      JOIN CUSTOMER_DIM C
             ON C.CUSTOMER_KEY = SF.CUSTOMER_KEY
      GROUP BY C.STATE,D.CAL_YEAR
      ORDER BY C.STATE;

CREATE OR REPLACE VIEW Prev_year_sales_Vw AS
      SELECT D.CAL_YEAR Year, SUM(Total_Amount) as YearSales
      FROM SALES_FACT SF
      JOIN DATE_DIM D
            ON SF.DATE_KEY = D.DATE_KEY
      GROUP BY  D.CAL_YEAR
      ORDER BY D.CAL_YEAR;

CREATE OR REPLACE VIEW best_in_year_Vw AS
      SELECT C.STATE,D.CAL_YEAR Year,
      SUM(Total_Amount) as TotalSales
      FROM SALES_FACT SF
      JOIN DATE_DIM D
            ON SF.DATE_KEY = D.DATE_KEY
      JOIN CUSTOMER_DIM C
            ON C.CUSTOMER_KEY = SF.CUSTOMER_KEY
      GROUP BY D.CAL_YEAR, C.State
      ORDER BY TotalSales ;


CREATE OR REPLACE PROCEDURE AnnualStateSalesGrowth(curr_year IN NUMBER) IS

      v_state         varchar(50);
      v_year          State_YoY_Vw.Year%TYPE;
      v_previousYear        State_YoY_Vw.PreviousYear%TYPE;
      v_prevYearTotal       State_YoY_Vw.PastYearTotal%TYPE;
      v_YoY           State_YoY_Vw.YoY%TYPE;
      v_yearTotal     State_YoY_Vw.YearTotal%TYPE;
      v_totalSales    NUMBER (15) := 0;
      v_yearSales     NUMBER (15) := 0;
      v_numState      NUMBER(3) := 0;
      v_startYear     NUMBER(4) := curr_year-3;
      v_midYear       NUMBER(4) := curr_year-2;
      v_endYear       NUMBER(4) := curr_year-1;
      v_counter       NUMBER(1) := 0;
      v_3_sales       NUMBER(15) := 0;
      v_2_sales       NUMBER(15) := 0;
      v_1_sales       NUMBER(15) := 0;
      v_location      varchar(50) := 0;

      CURSOR state_list_cur IS
            SELECT C.State,
            SUM(Total_Amount) totalSales
            FROM SALES_FACT SF
            JOIN DATE_DIM D
                  ON SF.DATE_KEY = D.DATE_KEY
            JOIN CUSTOMER_DIM C
                  ON C.CUSTOMER_KEY = SF.CUSTOMER_KEY
            GROUP BY C.State
            ORDER BY totalSales DESC;
```

```
CURSOR best_in_year_cur IS
      SELECT State,
      TotalSales
      FROM best_in_year_Vw
      WHERE Year = v_year;

CURSOR best_growth_in_year_cur IS
      SELECT State,YearTotal,PastYearTotal, YoY
      FROM State_YoY_Vw
      WHERE Year = v_year
      ORDER BY YoY DESC;

CURSOR worst_growth_in_year_cur IS
      SELECT State, YearTotal ,PastYearTotal, YoY
      FROM State_YoY_Vw
      WHERE Year = v_year
      ORDER BY YoY;

CURSOR state_YoY_cur IS
      SELECT state,Year, YearTotal,PreviousYear,PastYearTotal, YoY
      FROM State_YoY_Vw
      WHERE state = v_state
      AND (Year = v_startYear
      OR Year = v_midYear
      OR Year = v_endYear);

CURSOR Best_In_3_cur IS
      SELECT state, TotalSales
      FROM (
            SELECT C.STATE, SUM(Total_Amount) TotalSales
            FROM SALES_FACT SF
            JOIN DATE_DIM D
                  ON SF.DATE_KEY = D.DATE_KEY
            JOIN CUSTOMER_DIM C
                  ON SF.CUSTOMER_KEY = C.CUSTOMER_KEY
            WHERE D.CAL_YEAR = v_startYear
            OR D.CAL_YEAR = v_midYear
            OR D.CAL_YEAR = v_endYear
            GROUP BY C.STATE
            ORDER BY TotalSales DESC
      )
      WHERE STATE = v_state;

CURSOR prev3_sales_cur IS
      SELECT YearSales
      FROM (
            Prev_year_sales_Vw
      )
      WHERE Year = v_startYear;

CURSOR prev2_sales_cur IS
      SELECT YearSales
      FROM (
            Prev_year_sales_Vw
      )
```

```
            WHERE Year = v_midYear;

      CURSOR prev1_sales_cur IS
            SELECT YearSales
            FROM (
                  Prev_year_sales_Vw
            )
            WHERE Year = v_endYear;

      CURSOR total_sales IS
            SELECT SUM(YearSales) as Sales
            FROM (
                  Prev_year_sales_Vw
            )
            WHERE Year = v_startYear
            OR Year = v_midYear
            OR Year = v_endYear;



BEGIN
      -- Get State
      SELECT COUNT(DISTINCT State) INTO  v_numState
      FROM SALES_FACT SF
      JOIN DATE_DIM D
            ON SF.DATE_KEY = D.DATE_KEY
      JOIN CUSTOMER_DIM C
            ON C.CUSTOMER_KEY = SF.CUSTOMER_KEY;

      -- Outputs
      DBMS_OUTPUT.PUT_LINE(LPAD('=', 139, '='));
      DBMS_OUTPUT.PUT_LINE(' | '||LPAD('Computer Product
      Supplier', 80, ' ')||LPAD(' | ', 58, ' '));
      DBMS_OUTPUT.PUT_LINE(' | '||LPAD('Annual Sales Growth by
      State From ', 75)||v_startYear||LPAD(' To ',
      4)||v_endYear||LPAD(' | ', 51, ' '));
      DBMS_OUTPUT.PUT_LINE(' | '||RPAD('YoY = Year on Year',
      135)||' | ');
      DBMS_OUTPUT.PUT_LINE(' | '||LPAD('Number of States : ',
      19)||RPAD(v_numState, 3)||LPAD('Report Generated: ',
      94)||TO_CHAR(SYSDATE, 'DD-MM-YYYY HH24:MI:SS')||' | ');
      DBMS_OUTPUT.PUT_LINE(LPAD('-', 139, '-'));
      DBMS_OUTPUT.PUT_LINE(RPAD(' |      State     ',24)||RPAD('
      |     Year ',20)||
      RPAD(' |   YoY  ',10)||RPAD(' |    Year ',19)||
      RPAD(' |   YoY   ',11)||RPAD(' |     Year ',19)||RPAD(' |
      YoY  ',11)||
      RPAD(' ||   Total  ',18)||LPAD(' | ',9));
      DBMS_OUTPUT.PUT_LINE(RPAD(' |           ',25)||'|  '||
      RPAD(TO_CHAR(v_startYear),7)||'(RM)      '||'|  '||
      RPAD(TO_CHAR(v_startYear),6)||' |  '||
      RPAD(TO_CHAR(v_midYear),7)||'(RM)     '||'|  '||
      RPAD(TO_CHAR(v_midYear),5)||'   | '||
      RPAD(TO_CHAR(v_endYear),7)||'(RM)      '||'|  '||
      RPAD(TO_CHAR(v_endYear),7)||
      RPAD(' ||   (RM)   ',14)||LPAD(' | ',13));
```

```
DBMS_OUTPUT.PUT_LINE(LPAD('-', 139, '-'));

OPEN state_list_cur;
FETCH state_list_cur INTO v_state, v_totalSales;
WHILE(state_list_cur%FOUND) LOOP
      OPEN state_YoY_cur;
      FETCH state_YoY_cur INTO
            v_state, v_year, v_yearTotal, v_previousYear,
            v_prevYearTotal,v_YoY;
      DBMS_OUTPUT.PUT(' |  '||RPAD(v_state, 20)||' | ');
      WHILE (state_YoY_cur%FOUND) LOOP
            DBMS_OUTPUT.PUT(TO_CHAR(v_yearTotal,
            '$9,999,999,999')||'   |');
            IF (v_year = v_startYear)
            THEN
                  DBMS_OUTPUT.PUT(RPAD('  --  ', 6)||'%  |');
            ELSE
                  DBMS_OUTPUT.PUT('  '||RPAD(TO_CHAR(v_YoY,
                  '90.99' ),6)||'% |');
            END IF;
            FETCH state_YoY_cur INTO
                  v_state, v_year, v_yearTotal,
                  v_previousYear,v_prevYearTotal, v_YoY;
      END LOOP;
      OPEN Best_In_3_cur;
      FETCH Best_In_3_cur INTO v_state, v_totalSales;
      DBMS_OUTPUT.PUT('|'||RPAD(to_char(v_totalSales,
      '$9,999,999,999'), 21)||' |');
      CLOSE state_YoY_cur;
      CLOSE Best_In_3_cur;

FETCH state_list_cur INTO v_state, v_totalSales;
DBMS_OUTPUT.PUT_LINE(chr(001));
END LOOP;

OPEN prev3_sales_cur;
FETCH prev3_sales_cur INTO v_3_sales;
OPEN prev2_sales_cur;
FETCH prev2_sales_cur INTO v_2_sales;
OPEN prev1_sales_cur;
FETCH prev1_sales_cur INTO v_1_sales;
OPEN total_sales;
FETCH total_sales INTO v_totalSales;
DBMS_OUTPUT.PUT_LINE(LPAD('-', 139, '-'));

DBMS_OUTPUT.PUT_LINE(' | '||LPAD('Total Sales :',21)||' |
' ||' ' ||
RPAD(to_char(v_3_sales, '$999,999,999'), 15)||' | '
||LPAD('        | ',9) ||'  '||
RPAD(to_char(v_2_sales, '$999,999,999'), 16)||LPAD('|
| ',12) ||'  '||
RPAD(to_char(v_1_sales, '$999,999,999'), 16)||'| ' ||
LPAD('        ||',11) ||
RPAD(to_char(v_totalSales, '$9,999,999,999'), 16)|| '
|');
```

```
DBMS_OUTPUT.PUT_LINE(LPAD('-', 139, '-'));
DBMS_OUTPUT.PUT_LINE(' | '||LPAD ('Summary',77,' ')||
LPAD(' | ', 61));
DBMS_OUTPUT.PUT_LINE(' | '||LPAD ('============',80,' ')||
LPAD(' | ', 58));

v_year := v_startYear;

WHILE (v_year <> v_endYear+1) LOOP
OPEN best_in_year_cur;
FETCH best_in_year_cur INTO v_state, v_totalSales;

v_location := CONCAT(TO_CHAR(v_state),',');

--DBMS_OUTPUT.PUT_LINE(' | '||v_year||' > '||RPAD('Best
Sales : ',18)||
--RPAD(' State ',8)||': '||RPAD(v_location,35)||' | Sales :
'||
--RPAD(TO_CHAR(v_totalSales, '$99,999,999.99'), 30)||
--LPAD(' | ', 22));

IF (v_year <> v_startyear) THEN

-- Highest Growth
OPEN best_growth_in_year_cur;
FETCH best_growth_in_year_cur INTO v_state, v_yearTotal
,v_prevYearTotal, v_YoY;

v_location := CONCAT(TO_CHAR(v_state),',');

DBMS_OUTPUT.PUT_LINE(' | '||v_year||' > '||RPAD('Highest
Growth : ', 18)||
RPAD('  State',8)||' :  '||RPAD(v_location, 35)||'  | YoY :
'||
RPAD(TO_CHAR(v_YoY), 6)||RPAD('%',4)||RPAD('From ',5)||
RPAD(TO_CHAR(v_prevYearTotal, '$999,999,999.99'),
15)||RPAD(' To ',3)||
RPAD(TO_CHAR(v_yearTotal, '$999,999,999.99'), 15)||LPAD(' |
', 8));


-- Lowest Growth
OPEN worst_growth_in_year_cur;
FETCH worst_growth_in_year_cur INTO v_state, v_yearTotal
,v_prevYearTotal, v_YoY;

v_location := CONCAT(TO_CHAR(v_state),',');

DBMS_OUTPUT.PUT_LINE(' | '||'    '||' > '||RPAD('Worst
Growth    ', 15)||':   '||
RPAD(' State',7)||' :  '||RPAD(v_location, 35)||'  | YoY :
'||
RPAD(TO_CHAR(v_YoY), 6)||RPAD('%',4)||RPAD('From ',5)||
RPAD(TO_CHAR(v_prevYearTotal, '$999,999,999.99'),
15)||RPAD(' To ',3)||
RPAD(TO_CHAR(v_yearTotal, '$999,999,999.99'), 15)||'      |
');
```

```
        CLOSE best_growth_in_year_cur;
        CLOSE worst_growth_in_year_cur;

        END IF;
        CLOSE best_in_year_cur;
        v_year := v_year + 1;
        DBMS_OUTPUT.PUT_LINE(LPAD('-', 139, '-'));
        END LOOP;

        -- End of Report
        DBMS_OUTPUT.PUT_LINE(LPAD ('-', 139, '-'));
        DBMS_OUTPUT.PUT_LINE(' | '||LPAD ('END OF REPORT',80,'
        ')||LPAD(' | ', 58));
        DBMS_OUTPUT.PUT_LINE(LPAD('=', 139, '='));

END;
/

EXEC AnnualStateSalesGrowth(2022);
```

```
SQL> EXEC AnnualStateSalesGrowth(2022);
```

Computer Product Supplier

Annual Sales Growth by State From 2019 To 2021

YoY = Year on Year

Number of States : 13

Report Generated: 30-04-2023 11:56:45

State	Year 2019 (RM)	YoY 2019	Year 2020 (RM)	YoY 2020	Year 2021 (RM)	YoY 2021	Total (RM)
Selangor	$205,475,584	-- %	$204,320,122	-0.56%	$206,229,613	0.93%	$616,025,319
Wilayah Persekutuan	$80,418,188	-- %	$80,429,948	0.01%	$78,807,883	-2.02%	$239,656,019
Sarawak	$41,710,084	-- %	$42,460,493	1.80%	$40,957,349	-3.54%	$125,127,926
Pulau Pinang	$39,573,237	-- %	$39,435,317	-0.35%	$39,882,356	1.13%	$118,890,910
Sabah	$16,935,130	-- %	$17,806,614	5.15%	$17,678,836	-0.72%	$52,420,580
Pahang	$13,888,051	-- %	$13,920,125	0.23%	$13,653,050	-1.92%	$41,461,225
Perak	$8,709,548	-- %	$8,337,151	-4.28%	$8,732,799	4.75%	$25,779,497
Perlis	$5,680,805	-- %	$5,290,248	-6.88%	$5,203,330	-1.64%	$16,174,383
Negeri Sembilan	$4,523,555	-- %	$4,726,670	4.49%	$4,976,042	5.28%	$14,226,266
Kedah	$2,566,114	-- %	$2,629,114	2.46%	$2,545,037	-3.20%	$7,740,264
Melaka	$1,697,592	-- %	$1,794,457	5.71%	$1,704,430	-5.02%	$5,196,478
Johor	$1,701,350	-- %	$1,767,127	3.87%	$1,769,512	0.13%	$5,237,990
Kelantan	$1,388,390	-- %	$1,533,186	10.43%	$1,293,982	-15.60%	$4,215,558
Total Sales :	$424,267,627		$424,450,570		$423,434,218		$1,272,152,415

Summary

2020 > Highest Growth :	State :	Kelantan,	YoY : 10.43 %	From	$1,388,390.0	To $1,533,185.6
> Worst Growth :	State :	Perlis,	YoY : -6.88 %	From	$5,680,805.0	To $5,290,248.1
2021 > Highest Growth :	State :	Negeri Sembilan,	YoY : 5.28 %	From	$4,726,669.5	To $4,976,041.5
> Worst Growth :	State :	Kelantan,	YoY : -15.6 %	From	$1,533,185.6	To $1,293,981.7

END OF REPORT

The data generated by this report is useful for analyzing trends. For instance, Negeri Sembilan showed the highest YoY growth rate in 2021 at 5.28%. This information can be used by the company to increase the distribution of product lines in Negeri Sembilan and allocate more resources, such as additional employees, to the state. Although Selangor is currently the best-performing state based on the report's findings, its YoY growth rate in 2021 was only 0.93%, which is significantly lower than Negeri Sembilan's YoY growth rate. Therefore, some employees can be relocated from Selangor to Negeri Sembilan, as it is growing at a faster pace and may eventually outperform Selangor.