FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

# BACS3183 ADVANCED DATABASE MANAGEMENT

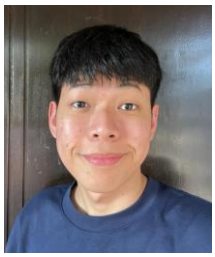
Assignment

Semester 202209

| | | |
|---|---|---|
| Programme (Year & Group) | : | RDS2S1, RSD3S4 |
| Tutorial Group | : | G3 |
| Date Submitted | : | 26 Dec 2022 |

Team members:

| No | Name (Block Letters) | Registration No. | Signature | Marks |
|---|---|---|---|---|
| 1 | TANG SHARREN | 21WMR01086 | sharren | **91.5** |
| 2 | TONG ZIAN CHUAN | 20WMR09404 | **chuan** | **79** |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |



Tang Sharren Tong Zian Chuan


Declaration

We confirm that we have read and shall comply with all the terms and conditions of TAR University College's plagiarism policy.

We declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own properly derived work.

| | | | | | | |
|---|---|---|---|---|---|---|
| Signature | : | sharren | **chuan** | | | |
| Name | : | TANG SHARREN | TONG ZIAN CHUAN | | | |
| Date | : | 26 Dec 2022 | 26 Dec 2022 | | | |

Assignment Assessment Form

Programme (Year-Semester-Group):

Member Name (Alphabetical order):

| |
|---|
| RDS2S1, RSD3S4 |
| 1. TANG SHARREN |
| 2.TONG ZIAN CHUAN |

| Task No. | Task Descriptions | Weightage | Criteria | 1 | 2 | 3 | 4 | 5 | Comment |
|---|---|---|---|---|---|---|---|---|---|
| 1 (CLO 1) | Entity Relationship Diagram | 10% | ● A complete ER data model in 3rd Normal Form.<br>● All primary keys, foreign keys, relationships and attributes must be clearly shown. | | | 9½ | | | |
| 2 (CLO 1) | Data Definition (DDL) | 10% | ● Relevant integrity constraints to ensure database integrity must be included.<br>● Necessary check constraints and default values to enforce business rules should also be included. | | | 9 | | | |
| 3 (CLO 1) | Data records | 5% | ● Sufficient quality data records must be created for each table. | | | 4 | | | |
| 4 (CLO 3) | Queries | 5% | ● Each team member is to design and produce two quality and useful queries for decision making at any two different management levels: strategic, tactical or operational.<br>● Single table queries are not allowed.<br>● | **5** | **3** | | | | |
| | | 5% | | **5** | **3** | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | ● Multiple table queries with aggregate functions (where appropriate) must be used.<br>● View(s) is/are ought to be incorporated, where necessary.<br>● These 2 queries cannot be used directly for report body.<br>● | | | | | | |
| 5<br>(CLO 2) | Stored Procedures | 5% | ● Each team member is to design and create two stored procedures that cater for the use case scenarios for the system.<br>● Quality, usefulness and importance of the stored procedures must be considered. | **5** | **3.5** | | | | |
| | | 5% | | **4** | **4** | | | | |
| 6<br>(CLO 2) | Triggers | 5% | ● Each team member is to design and create two triggers that enforces system-wide business rules and policies.<br>● Quality, usefulness and functionality of the triggers must be considered. | **5** | **3** | | | | |
| | | 5% | | **5** | **4** | | | | |
| 7<br>(CLO 3) | Reports | 10% | ● Each team member is to create two procedures to generate two reports (summary, detail and on demand basis reports) for the company.<br>● Parameter value(s) should be passed to the procedure, where necessary.<br>● Cursor must be used in report generation.<br>● Usefulness and presentation of the reports must be considered. | **9** | **9** | | | | |
| | | 10% | | **9** | **9** | | | | |
| 8<br>(CLO 2) | Extra Effort | 10% | ● Indexes, functions, views and/or user-defined exceptions must be incorporated where necessary.<br>● Usefulness and application of each of the above to enhance the efficiency and effectiveness of the information system must be considered. | **9** | **5** | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | ● Linking of all the tasks of every team member in creating a quality information system must be considered. | | | | | | |
| 9 (CLO 3) | Presentation & Participation | 15% | ● Run the single script file from Task 2 – 3 to create the new system database on the lab server.<br>● Individual presentation on Task 4 – 8.<br>● Q & A<br>● Actively participate in class discussion. | **13** | **13** | | | | |
| Assignment Marks / 100 | | | | **91.5** | **79** | | | | |

*CLO 1: Develop the relational database system with the appropriate integrity constraints and security control. (P3, PLO3)
*CLO 2: Design the solutions to issues pertaining to database efficiency and effectiveness using appropriate techniques. (C5, PLO2)
*CLO 3: Extract information from the database using efficient SQL query construct. (C4, PLO6)

**Table of Content**

# Chapter 1 Background of the System

## 1.1 Introduction

Rush Hour Mart is a national-wide mini mart with 10 branches located across the peninsular of Malaysia. This system enables the company to keep records of its transactions. There is 300 staffs working in the company and many transactions are made every day, so it is crucial to organize the data. This system allows the insertion, modification, and deletion of the company's database by the manager with the highest authority. This system will also be able to produce a report or summary like a monthly sales report or find high-demand products which will benefit the company.

## 1.2 Entities of System

| **Base (Parent) Table** | **Transaction (Child) Table** | **Associative (Bridge) Table** |
|---|---|---|
| Customer<br>Member<br>Product<br>Branch<br>Supplier | Order<br>Staff<br>ShiftAllocation | OrderDetails<br>Supply |

# Chapter 2 Entity-Relationship Modeling

## 2.1 Business Rules and Assumptions

**Business Rules**

1. Each customer can have more than one order. An order can only belong to one customer (one-to-many).

2. Each order consists of one or many products, and a product can be included in one or many orders. (many-to-many)

3. Each customer may or may not be registered as a member, so a member must be a customer before registering as a member. (one-to-zero or one)

2. Each staff will handle one or many orders (one-to-many) so each order will only be handled by one staff. (one-to-one)

3. Each staff will be allocated to one or many shifts, and each shift can be handled by one or many staffs. (many-to-many)

4. Each staff can only work at one branch. Each branch can have more than one staff. (one-to-many)

5. Each staff may or may not handle many supplies so a supply will only be handled by a staff(one-to-zero or many)

6. Each branch's products are sourced from many supplies(one-to-many)

7. Each product can be sourced from one or many supplies. (one-to-many)
Each supplier can supply more than one product. (many-to-many).

8. Each supplier will be contributed to one or many supplies(one-to-many)

**Assumptions**

1. The price sold of all products is the price supplied by the supplier.

2. The supermarket is open for business from 7 am to 10 pm

3. All customer are required to join as a member and only customer aged above 18 is allowed to join

4. An order record can be added and modified but cannot be deleted, however, the product records in the order record can be added, modified, and deleted.

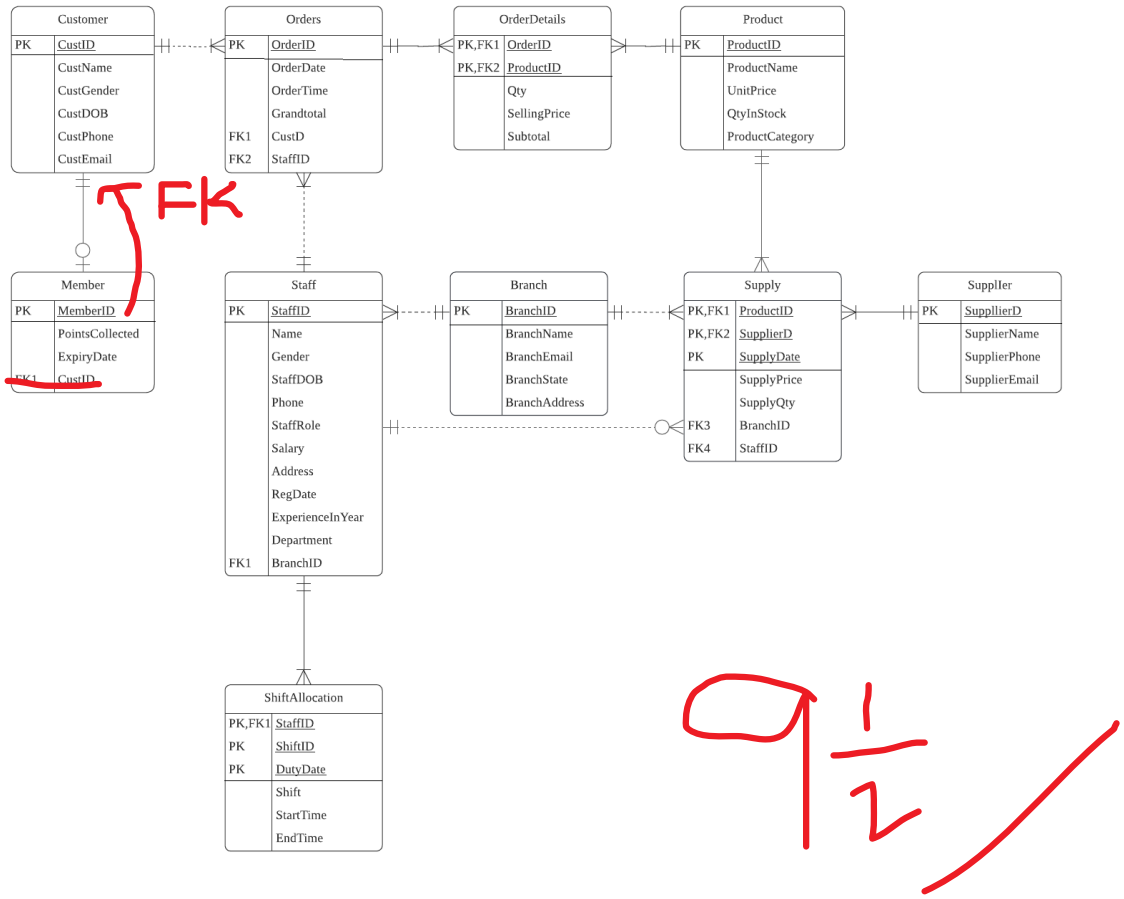
5. A staff record can be added and modified but cannot be removed.

6. Customers can request an order refund but that order cannot exceed 3 days and the total amount cannot exceed 300.

7.All staffs must be at least older than 18 years old.

8.In each branch, a department can only have a maximum number of 3 staffs with the same staff role.

## 2.2 ERD


Customer
PK CustID
CustName
CustGender
CustDOB
CustPhone
CustEmail
Orders
PK OrderID
OrderDate
OrderTime
Grandtotal
FK1 CustD
FK2 StaffID
OrderDetails
PK,FK1 OrderID
PK,FK2 ProductID
Qty
SellingPrice
Subtotal
Product
PK ProductID
ProductName
UnitPrice
QtyInStock
ProductCategory
FK
Member
PK MemberID
PointsCollected
ExpiryDate
FK1 CustID
Staff
PK StaffID
Name
Gender
StaffDOB
Phone
StaffRole
Salary
Address
RegDate
ExperienceInYear
Department
FK1 BranchID
Branch
PK BranchID
BranchName
BranchEmail
BranchState
BranchAddress
Supply
PK,FK1 ProductID
PK,FK2 SupplierD
PK SupplyDate
SupplyPrice
SupplyQty
FK3 BranchID
FK4 StaffID
Supplier
PK SupplierD
SupplierName
SupplierPhone
SupplierEmail
ShiftAllocation
PK,FK1 StaffID
PK ShiftID
PK DutyDate
Shift
StartTime
EndTime

# Chapter 3 Data Definition

## 3.1 CREATE TABLE Statements

```
CREATE TABLE Supplier (
      SupplierID            VARCHAR(5)       NOT NULL,
      SupplierName     VARCHAR(30),
      SupplierPhone    VARCHAR(17),
      SupplierEmail    VARCHAR(30),
      PRIMARY KEY (SupplierID),
      CONSTRAINT chk_email_address CHECK
(REGEXP_LIKE(SupplierEmail,'^[a-zA-Z]\w+@(\S+)'))
);
CREATE TABLE Branch (
      BranchID         VARCHAR(5)       NOT NULL,
      BranchName            VARCHAR(30),
      BranchEmail           VARCHAR(30),
      BranchState      VARCHAR(17) DEFAULT 'Selangor',
      BranchAddress    VARCHAR(30),
      PRIMARY KEY (BranchID),
      CONSTRAINT chk_branchemail CHECK
(REGEXP_LIKE(BranchEmail,'^[a-zA-Z]\w+@(\S+)'))
);
CREATE TABLE Product (
      ProductID        VARCHAR(5)       NOT NULL,
      ProductName      VARCHAR(30),
      UnitPrice        NUMBER(4,2),
      QtyInStock         NUMBER(5),
      ProductCategory  VARCHAR(25),
      PRIMARY KEY (ProductID)
);

CREATE TABLE Customer (
      CustID     VARCHAR(5)       NOT NULL,
      CustName   VARCHAR(30),
      CustGender CHAR(1)          DEFAULT 'M',
      CustDOB    DATE,
      CustPhone  VARCHAR(20),
      CustEmail  VARCHAR(30),
      PRIMARY KEY (CustID),
      CONSTRAINT chk_custgender CHECK (UPPER(CustGender) IN
('M','F')),
      CONSTRAINT chk_custemail CHECK
(REGEXP_LIKE(CustEmail,'^[a-zA-Z]\w+@(\S+)'))
);
CREATE TABLE Member(
      MemberID         VARCHAR(5)       NOT NULL,
      PointsCollected  NUMBER(6)  DEFAULT 500,
      ExpiryDate         DATE,
      CustID           VARCHAR(5)       NOT NULL,
      PRIMARY KEY (MemberID),
      FOREIGN KEY (CustID) REFERENCES Customer (CustID)
);
CREATE TABLE Staff (
      StaffID          VARCHAR(5)       NOT NULL,
      Name             VARCHAR(20),
      Gender           CHAR(1)          DEFAULT 'M',
```

```
      StaffDOB         DATE,
      Phone            VARCHAR(17),
      StaffRole        VARCHAR(30)           DEFAULT 'Cleaning
Team',
      Salary           NUMBER(7,2)           DEFAULT 2000,
      Address          VARCHAR(26),
      RegDate          DATE,
      ExperienceInYear      NUMBER(1),
       Department      VARCHAR(26),
      BranchID         VARCHAR(5),
      PRIMARY KEY (StaffID),
      FOREIGN KEY (BranchID) REFERENCES Branch (BranchID),
      CONSTRAINT chk_staff_staffsex CHECK (UPPER(Gender) IN
('M','F'))
);
CREATE TABLE Orders (
      OrderID    VARCHAR(5) NOT NULL,
      OrderDate  DATE       NOT NULL,
      OrderTime      CHAR(8),
      Grandtotal  NUMBER(7,2),
      CustID     VARCHAR(4) NOT NULL,
      StaffID    VARCHAR(5) NOT NULL,
      PRIMARY KEY (OrderID),
      FOREIGN KEY (CustID) REFERENCES Customer (CustID),
      FOREIGN KEY (StaffID) REFERENCES Staff (StaffID)
);

CREATE TABLE ShiftAllocation (
      StaffID    VARCHAR(5)       NOT NULL,
      ShiftID    VARCHAR(6)       NOT NULL,
      DutyDate   DATE,
      StartTime  CHAR(8),
      EndTime    CHAR(8),
      PRIMARY KEY (StaffID,ShiftID,DutyDate),
      FOREIGN KEY (StaffID) REFERENCES Staff (StaffID)
);
CREATE TABLE Supply (
      SupplyDate       DATE              NOT NULL,
      SupplyQty  NUMBER(5),
      SupplyPrice NUMBER(7,2),
      ProductID  VARCHAR(4)       NOT NULL,
      SupplierID       VARCHAR(4)       NOT NULL,
      BranchID   VARCHAR(4)       NOT NULL,
       StaffID         VARCHAR(5)       NOT NULL,
      PRIMARY KEY (SupplyDate,ProductID,SupplierID),
      FOREIGN KEY (ProductID) REFERENCES Product (ProductID),
      FOREIGN KEY (SupplierID) REFERENCES Supplier
(SupplierID),
      FOREIGN KEY (BranchID) REFERENCES Branch (BranchID),
      FOREIGN KEY (StaffID) REFERENCES Staff (StaffID)
);
CREATE TABLE OrderDetails (
      OrderID          VARCHAR(5) NOT NULL,
      ProductID        VARCHAR(5) NOT NULL,
      Qty              NUMBER(4),
```

```
      SellingPrice     NUMBER(5,2),
      Subtotal         NUMBER(7,2),
      PRIMARY KEY (OrderID,ProductID),
      FOREIGN KEY (OrderID) REFERENCES Orders (OrderID),
      FOREIGN KEY (ProductID) REFERENCES Product (ProductID)
);
```

## 3.2 Sample Records (10 sample Records for each table)

### 3.2.1 Supplier

```
insert into Supplier (SupplierID , SupplierName,
SupplierPhone, SupplierEmail ) values ('SP1', 'Dabjam', '+66
430 220 5167', 'ddick0@paginegialle.it');

insert into Supplier (SupplierID , SupplierName,
SupplierPhone, SupplierEmail ) values ('SP2', 'Roodel', '+52
920 429 4843', 'lgrewcock1@youku.com');

insert into Supplier (SupplierID , SupplierName,
SupplierPhone, SupplierEmail ) values ('SP3', 'Rhyloo', '+420
615 912 0431', 'ndonne2@yellowbook.com');

insert into Supplier (SupplierID , SupplierName,
SupplierPhone, SupplierEmail ) values ('SP4', 'Zazio', '+212
764 211 8768', 'dwatkins3@walmart.com');

insert into Supplier (SupplierID , SupplierName,
SupplierPhone, SupplierEmail ) values ('SP5', 'Ailane', '+254
205 788 7967', 'ctremathack4@histats.com');

insert into Supplier (SupplierID , SupplierName,
SupplierPhone, SupplierEmail ) values ('SP6', 'Buzzshare',
'+33 468 612 8578', 'mgrinaway5@list-manage.com');

insert into Supplier (SupplierID , SupplierName,
SupplierPhone, SupplierEmail ) values ('SP7', 'Izio', '+689
728 498 0561', 'csaltmarsh6@smugmug.com');

insert into Supplier (SupplierID , SupplierName,
SupplierPhone, SupplierEmail ) values ('SP8', 'Rhycero', '+62
429 399 7774', 'uclawley7@nyu.edu');

insert into Supplier (SupplierID , SupplierName,
SupplierPhone, SupplierEmail ) values ('SP9', 'Camido', '+27
977 384 0173', 'rlyford8@people.com.cn');

insert into Supplier (SupplierID , SupplierName,
SupplierPhone, SupplierEmail ) values ('SP10', 'Photofeed',
'+62 429 569 4886', 'ioglessane9@army.mil');
```

### 3.2.2 Branch

```
insert into Branch (BranchID, BranchName, BranchEmail,
BranchState, BranchAddress) values ('B1', 'Serdang Raya',
'eatyea0@ucla.edu', 'Selangor', '7146 Westridge Alley');

insert into Branch (BranchID, BranchName, BranchEmail,
BranchState, BranchAddress) values ('B2', 'Danau Kota',
```

```
'sbritcher1@constantcontact.com', 'Kuala Lumpur', '31 Melrose
Court');

insert into Branch (BranchID, BranchName, BranchEmail,
BranchState, BranchAddress) values ('B3', 'Taming Jaya',
'gdanelut2@patch.com', 'Selangor', '237 Menomonie Hill');

insert into Branch (BranchID, BranchName, BranchEmail,
BranchState, BranchAddress) values ('B4', 'Pavilion',
'fbradberry3@ftc.gov', 'Selangor', '58 Buena Vista Lane');

insert into Branch (BranchID, BranchName, BranchEmail,
BranchState, BranchAddress) values ('B5', 'Lalaport',
'rtommis4@yahoo.com', 'Kuala Lumpur', '22645 Scoville Park');

insert into Branch (BranchID, BranchName, BranchEmail,
BranchState, BranchAddress) values ('B6', 'Amerin Mall',
'hgudeman5@ucoz.com', 'Selangor', '75 Russell Junction');

insert into Branch (BranchID, BranchName, BranchEmail,
BranchState, BranchAddress) values ('B7', 'Palm Mall',
'cbucklee6@goo.ne.jp', 'Seremban', '1794 Fulton Way');

insert into Branch (BranchID, BranchName, BranchEmail,
BranchState, BranchAddress) values ('B8', 'East Coast',
'rgoude7@google.co.jp', 'Pahang', '418 Jay Road');

insert into Branch (BranchID, BranchName, BranchEmail,
BranchState, BranchAddress) values ('B9', 'Tali Air',
'alevensky8@eepurl.com', 'Pahang', '8 Shelley Terrace');

insert into Branch (BranchID, BranchName, BranchEmail,
BranchState, BranchAddress) values ('B10', 'Puncak Jalil',
'tdruel9@sphinn.com', 'Selangor', '283 Luster Point');
```

### 3.2.3 Product

```
insert into Product (ProductID, ProductName, UnitPrice,
QtyInStock, ProductCategory) values ('P1', 'Chinese Lemon
Pork', 9.66, 1000, 'Dried Food');

insert into Product (ProductID, ProductName, UnitPrice,
QtyInStock, ProductCategory) values ('P2', 'Garlic - Primerba,
Paste', 17.63, 3600, 'Fruits');

insert into Product (ProductID, ProductName, UnitPrice,
QtyInStock, ProductCategory) values ('P3', 'Bag Stand', 6.14,
850, 'Cooking Condiments');

insert into Product (ProductID, ProductName, UnitPrice,
QtyInStock, ProductCategory) values ('P4', 'Wasabi Paste',
17.49, 820, 'Vegetable');

insert into Product (ProductID, ProductName, UnitPrice,
QtyInStock, ProductCategory) values ('P5', 'Coffee -
Decaffeinato Coffee', 19.33, 580, 'Fresh Produce');
```

```
insert into Product (ProductID, ProductName, UnitPrice,
QtyInStock, ProductCategory) values ('P6', 'Oats Large Flake',
10.13, 280, 'Fresh Produce');

insert into Product (ProductID, ProductName, UnitPrice,
QtyInStock, ProductCategory) values ('P7', 'Syrup - Pancake',
15.26, 1400, 'Vegetable');

insert into Product (ProductID, ProductName, UnitPrice,
QtyInStock, ProductCategory) values ('P8', 'Chocolate Bar -
Smarties', 18.10, 330, 'Vegetable');

insert into Product (ProductID, ProductName, UnitPrice,
QtyInStock, ProductCategory) values ('P9', 'Lemon Balm -
Fresh', 3.78, 650, 'Cooking Condiments');

insert into Product (ProductID, ProductName, UnitPrice,
QtyInStock, ProductCategory) values ('P10', 'Vinegar -
Raspberry', 14.66, 940, 'Fruits');
```

### 3.2.4 Customer

```
insert into Customer (CustID, CustName, CustGender, CustDOB,
CustPhone, CustEmail) values ('CU1', 'Ridger', 'F',
'03/08/1993', '+33 723 965 4213', 'kridger0@webs.com');

insert into Customer (CustID, CustName, CustGender, CustDOB,
CustPhone, CustEmail) values ('CU2', 'Vedekhov', 'M',
'12/12/1997', '+351 712 530 8685', 'vvedekhov1@aol.com');

insert into Customer (CustID, CustName, CustGender, CustDOB,
CustPhone, CustEmail) values ('CU3', 'Kuhnwald', 'M',
'24/07/1994', '+30 844 248 3023',
'kkuhnwald2@cloudflare.com');

insert into Customer (CustID, CustName, CustGender, CustDOB,
CustPhone, CustEmail) values ('CU4', 'Sides', 'F',
'25/03/2003', '+63 925 450 9688', 'jsides3@ow.ly');

insert into Customer (CustID, CustName, CustGender, CustDOB,
CustPhone, CustEmail) values ('CU5', 'Madders', 'M',
'16/02/1994', '+86 142 857 6290', 'zmadders4@unesco.org');

insert into Customer (CustID, CustName, CustGender, CustDOB,
CustPhone, CustEmail) values ('CU6', 'Hoyes', 'F',
'08/07/1988', '+503 895 232 9437', 'khoyes5@sbwire.com');

insert into Customer (CustID, CustName, CustGender, CustDOB,
CustPhone, CustEmail) values ('CU7', 'Tuffell', 'F',
'23/02/2000', '+62 870 315 8602', 'ntuffell6@yandex.ru');

insert into Customer (CustID, CustName, CustGender, CustDOB,
CustPhone, CustEmail) values ('CU8', 'Ruos', 'F',
'16/01/1995', '+261 259 258 8819', 'lruos7@t.co');

insert into Customer (CustID, CustName, CustGender, CustDOB,
CustPhone, CustEmail) values ('CU9', 'Beeton', 'F',
'17/04/1986', '+7 764 770 2214', 'rbeeton8@gmpg.org');
```

```
insert into Customer (CustID, CustName, CustGender, CustDOB,
CustPhone, CustEmail) values ('CU10', 'Anthoine', 'F',
'13/04/1998', '+82 893 871 9813',
'santhoine9@thetimes.co.uk');
```

### 3.2.5 Member

```
insert into Member (MemberID, PointsCollected, ExpiryDate,
CustID) values ('M1', 6682, '27/01/2024', 'CU3');

insert into Member (MemberID, PointsCollected, ExpiryDate,
CustID) values ('M2', 2497, '13/06/2023', 'CU1');

insert into Member (MemberID, PointsCollected, ExpiryDate,
CustID) values ('M3', 8402, '04/12/2023', 'CU7');

insert into Member (MemberID, PointsCollected, ExpiryDate,
CustID) values ('M4', 8546, '27/10/2023', 'CU5');

insert into Member (MemberID, PointsCollected, ExpiryDate,
CustID) values ('M5', 6548, '30/05/2023', 'CU6');

insert into Member (MemberID, PointsCollected, ExpiryDate,
CustID) values ('M6', 4903, '09/09/2023', 'CU9');

insert into Member (MemberID, PointsCollected, ExpiryDate,
CustID) values ('M7', 5183, '08/06/2023', 'CU4');

insert into Member (MemberID, PointsCollected, ExpiryDate,
CustID) values ('M8', 3191, '14/04/2024', 'CU10');

insert into Member (MemberID, PointsCollected, ExpiryDate,
CustID) values ('M9', 3991, '26/07/2023', 'CU2');

insert into Member (MemberID, PointsCollected, ExpiryDate,
CustID) values ('M10', 9145, '15/02/2024', 'CU8');
```

### 3.2.6 Staff

```
insert into Staff (StaffID, Name, Gender, StaffDOB, Phone,
StaffRole, Salary, Address, RegDate, ExperienceInYear,
Department, BranchID) values ('ST1', 'Web Seys', 'F',
'21/02/2022', '+63 804 335 5222', 'Stock clerks', 3748, '165
Magdeline Drive', '28/04/2018', 8, 'Accounting and Finance',
'B10');

insert into Staff (StaffID, Name, Gender, StaffDOB, Phone,
StaffRole, Salary, Address, RegDate, ExperienceInYear,
Department, BranchID) values ('ST2', 'Ferdinand Sturt', 'F',
'20/04/2022', '+1 321 553 2269', 'Product Buyer', 4372, '0476
Southridge Court', '26/08/2015', 5, 'Human Resources', 'B7');

insert into Staff (StaffID, Name, Gender, StaffDOB, Phone,
StaffRole, Salary, Address, RegDate, ExperienceInYear,
Department, BranchID) values ('ST3', 'Ebeneser Janus', 'F',
'21/10/2022', '+1 763 710 4612', 'Department Manager', 3622,
'5429 Merchant Junction', '20/06/2019', 3, 'Marketing', 'B3');
```

```
insert into Staff (StaffID, Name, Gender, StaffDOB, Phone,
StaffRole, Salary, Address, RegDate, ExperienceInYear,
Department, BranchID) values ('ST4', 'Ty Borzone', 'M',
'16/01/2022', '+86 616 201 9725', 'Stock clerks', 4529, '66500
Buell Court', '15/11/2017', 8, 'Human Resources', 'B8');

insert into Staff (StaffID, Name, Gender, StaffDOB, Phone,
StaffRole, Salary, Address, RegDate, ExperienceInYear,
Department, BranchID) values ('ST5', 'Cordell Kos', 'M',
'11/12/2021', '+7 672 663 2279', 'Wine experts', 4434, '5990
Mitchell Way', '01/02/2016', 4, 'Production', 'B5');

insert into Staff (StaffID, Name, Gender, StaffDOB, Phone,
StaffRole, Salary, Address, RegDate, ExperienceInYear,
Department, BranchID) values ('ST6', 'Mathilda Marzella', 'M',
'27/12/2021', '+63 321 565 1915', 'Shipping and receiving
clerks', 2350, '02227 Loomis Court', '28/12/2020', 8, 'Human
Resources', 'B9');

insert into Staff (StaffID, Name, Gender, StaffDOB, Phone,
StaffRole, Salary, Address, RegDate, ExperienceInYear,
Department, BranchID) values ('ST7', 'Brett Twycross', 'F',
'18/02/2022', '+62 196 380 1622', 'Baggers', 2843, '6151
Bobwhite Street', '22/11/2021', 8, 'Marketing', 'B2');

insert into Staff (StaffID, Name, Gender, StaffDOB, Phone,
StaffRole, Salary, Address, RegDate, ExperienceInYear,
Department, BranchID) values ('ST8', 'Trude Hinzer', 'F',
'15/01/2022', '+261 540 130 4088', 'Store manager', 2214,
'6665 Holy Cross Trail', '04/09/2018', 4, 'Accounting and
Finance', 'B1');

insert into Staff (StaffID, Name, Gender, StaffDOB, Phone,
StaffRole, Salary, Address, RegDate, ExperienceInYear,
Department, BranchID) values ('ST9', 'Ethe Heavy', 'F',
'06/06/2022', '+967 713 893 0859', 'Wine experts', 4031, '948
Kipling Hill', '05/07/2021', 3, 'Marketing', 'B6');

insert into Staff (StaffID, Name, Gender, StaffDOB, Phone,
StaffRole, Salary, Address, RegDate, ExperienceInYear,
Department, BranchID) values ('ST10', 'Fidelia Jovis', 'F',
'23/11/2022', '+420 588 554 4608', 'Meat cutters', 4688, '308
Dayton Parkway', '01/02/2020', 5, 'Research and Development',
'B4');
```

### 3.2.7 Orders

```
insert into Orders (OrderID, OrderDate, OrderTime,
Grandtotal, CustID, StaffID) values ('OR1', '07/09/2017',
'6:44 PM', 900, 'CU6', 'ST55');
insert into Orders (OrderID, OrderDate, OrderTime,
Grandtotal, CustID, StaffID) values ('OR2', '16/05/2016',
'10:23 AM', 35, 'CU10', 'ST2');
```

```
insert into Orders (OrderID, OrderDate, OrderTime,
Grandtotal, CustID, StaffID) values ('OR3', '29/09/2015',
'6:33 PM', 909, 'CU1', 'ST81');
insert into Orders (OrderID, OrderDate, OrderTime,
Grandtotal, CustID, StaffID) values ('OR4', '09/03/2021',
'2:31 PM', 431, 'CU7', 'ST6');
insert into Orders (OrderID, OrderDate, OrderTime,
Grandtotal, CustID, StaffID) values ('OR5', '17/07/2019',
'8:57 PM', 497, 'CU2', 'ST25');
insert into Orders (OrderID, OrderDate, OrderTime,
Grandtotal, CustID, StaffID) values ('OR6', '28/07/2016',
'2:59 PM', 681, 'CU8', 'ST42');
insert into Orders (OrderID, OrderDate, OrderTime,
Grandtotal, CustID, StaffID) values ('OR7', '24/10/2015',
'4:25 PM', 740, 'CU3', 'ST74');
insert into Orders (OrderID, OrderDate, OrderTime,
Grandtotal, CustID, StaffID) values ('OR8', '21/04/2016',
'10:52 AM', 220, 'CU4', 'ST29');
insert into Orders (OrderID, OrderDate, OrderTime,
Grandtotal, CustID, StaffID) values ('OR9', '22/12/2017',
'1:24 PM', 447, 'CU9', 'ST41');
insert into Orders (OrderID, OrderDate, OrderTime,
Grandtotal, CustID, StaffID) values ('OR10',
'09/03/2019', '12:34 PM', 976, 'CU5', 'ST51');
```

### 3.2.8 ShiftAllocation

```
insert into ShiftAllocation (StaffID, ShiftID, DutyDate,
StartTime, EndTime) values ('ST100', 'SHF1',
'08/01/2018', '9:48 AM', '9:02 AM');
insert into ShiftAllocation (StaffID, ShiftID, DutyDate,
StartTime, EndTime) values ('ST69', 'SHF2', '06/04/2020',
'6:16 PM', '11:25 AM');
insert into ShiftAllocation (StaffID, ShiftID, DutyDate,
StartTime, EndTime) values ('ST75', 'SHF3', '17/07/2022',
'8:53 PM', '5:50 PM');
insert into ShiftAllocation (StaffID, ShiftID, DutyDate,
StartTime, EndTime) values ('ST19', 'SHF4', '04/02/2017',
'4:27 PM', '6:21 PM');
insert into ShiftAllocation (StaffID, ShiftID, DutyDate,
StartTime, EndTime) values ('ST81', 'SHF5', '21/04/2020',
'4:58 PM', '7:14 PM');
insert into ShiftAllocation (StaffID, ShiftID, DutyDate,
StartTime, EndTime) values ('ST70', 'SHF6', '10/09/2016',
'1:42 PM', '5:45 PM');
insert into ShiftAllocation (StaffID, ShiftID, DutyDate,
StartTime, EndTime) values ('ST28', 'SHF7', '28/11/2020',
'12:53 PM', '7:19 PM');
```

```
insert into ShiftAllocation (StaffID, ShiftID, DutyDate,
StartTime, EndTime) values ('ST21', 'SHF8', '23/02/2022',
'8:29 PM', '5:37 PM');
insert into ShiftAllocation (StaffID, ShiftID, DutyDate,
StartTime, EndTime) values ('ST46', 'SHF9', '07/06/2019',
'11:40 AM', '12:25 PM');
insert into ShiftAllocation (StaffID, ShiftID, DutyDate,
StartTime, EndTime) values ('ST30', 'SHF10',
'15/01/2021', '7:24 PM', '8:02 PM');
```

### 3.2.9 Supply

```
insert into Supply (SupplyDate, SupplyQty, SupplyPrice,
ProductID, SupplierID, BranchID, StaffID) values
('04/10/2019', 637, 3408, 'P7', 'SP2', 'B8', 'ST15');
insert into Supply (SupplyDate, SupplyQty, SupplyPrice,
ProductID, SupplierID, BranchID, StaffID) values
('24/12/2019', 401, 2519, 'P5', 'SP5', 'B7', 'ST64');
insert into Supply (SupplyDate, SupplyQty, SupplyPrice,
ProductID, SupplierID, BranchID, StaffID) values
('14/11/2022', 300, 2640, 'P9', 'SP3', 'B10', 'ST6');
insert into Supply (SupplyDate, SupplyQty, SupplyPrice,
ProductID, SupplierID, BranchID, StaffID) values
('13/02/2019', 393, 3183, 'P3', 'SP10', 'B1', 'ST44');
insert into Supply (SupplyDate, SupplyQty, SupplyPrice,
ProductID, SupplierID, BranchID, StaffID) values
('06/07/2016', 389, 4705, 'P2', 'SP7', 'B5', 'ST63');
insert into Supply (SupplyDate, SupplyQty, SupplyPrice,
ProductID, SupplierID, BranchID, StaffID) values
('07/10/2022', 534, 3603, 'P1', 'SP9', 'B3', 'ST60');
insert into Supply (SupplyDate, SupplyQty, SupplyPrice,
ProductID, SupplierID, BranchID, StaffID) values
('11/08/2017', 543, 4198, 'P4', 'SP1', 'B6', 'ST50');
insert into Supply (SupplyDate, SupplyQty, SupplyPrice,
ProductID, SupplierID, BranchID, StaffID) values
('02/06/2016', 367, 4673, 'P10', 'SP4', 'B9', 'ST70');
insert into Supply (SupplyDate, SupplyQty, SupplyPrice,
ProductID, SupplierID, BranchID, StaffID) values
('03/05/2016', 456, 4631, 'P6', 'SP6', 'B4', 'ST100');
insert into Supply (SupplyDate, SupplyQty, SupplyPrice,
ProductID, SupplierID, BranchID, StaffID) values
('07/03/2019', 623, 2405, 'P8', 'SP8', 'B2', 'ST30');
```

### 3.2.10 OrderDetails

```
insert into OrderDetails (OrderID, ProductID, Qty,
SellingPrice, Subtotal) values ('OR5', 'P2', 7, 167,
253);
insert into OrderDetails (OrderID, ProductID, Qty,
SellingPrice, Subtotal) values ('OR64', 'P6', 3, 187,
473);
insert into OrderDetails (OrderID, ProductID, Qty,
SellingPrice, Subtotal) values ('OR43', 'P9', 7, 141,
649);
insert into OrderDetails (OrderID, ProductID, Qty,
SellingPrice, Subtotal) values ('OR20', 'P5', 10, 84,
794);
insert into OrderDetails (OrderID, ProductID, Qty,
SellingPrice, Subtotal) values ('OR54', 'P2', 4, 274,
240);
insert into OrderDetails (OrderID, ProductID, Qty,
SellingPrice, Subtotal) values ('OR95', 'P3', 3, 183,
641);
insert into OrderDetails (OrderID, ProductID, Qty,
SellingPrice, Subtotal) values ('OR28', 'P7', 8, 207,
183);
insert into OrderDetails (OrderID, ProductID, Qty,
SellingPrice, Subtotal) values ('OR8', 'P10', 9, 158,
187);
insert into OrderDetails (OrderID, ProductID, Qty,
SellingPrice, Subtotal) values ('OR33', 'P8', 9, 211,
270);
insert into OrderDetails (OrderID, ProductID, Qty,
SellingPrice, Subtotal) values ('OR3', 'P4', 2, 154,
172);
```

# Chapter 4 Queries, Procedures, Triggers and Reports

## 4.1 Tang Sharren

### 4.1.1 Query 1: Top 3 Branch Based On Sales Within Given Year

Purpose: List out the top 3 best performing branches and reward the branches with incentive

SQL statement:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';
SET LINESIZE 180
SET PAGESIZE 100

COLUMN BranchID FORMAT A15 HEADING 'Branch ID'
COLUMN BranchName FORMAT A25 HEADING 'Branch Name'
COLUMN BranchEmail FORMAT A35 HEADING 'Branch Email'
COLUMN BranchAddress FORMAT A30 HEADING 'Branch Address'
COLUMN Sales FORMAT $999,999.99 HEADING 'Total Sales'

PROMPT This query will generate top 3 branch based on sales with given
input as following
ACCEPT v_year char FORMAT 'A4' PROMPT 'Enter the year : '

CREATE OR REPLACE VIEW QUERY1
AS SELECT *
FROM(
SELECT B.BranchID, BranchName,BranchEmail,BranchAddress,SUM(Grandtotal)
AS Sales
FROM Branch B, Staff S, Orders O
WHERE B.BranchID = S.BranchID
AND S.StaffID = O.StaffID
AND EXTRACT(YEAR FROM OrderDate) = '&v_year'
GROUP BY B.BranchID,BranchName,BranchEmail,BranchAddress
ORDER BY SUM(Grandtotal) DESC)
WHERE ROWNUM < 4;

COMPUTE SUM LABEL 'SUM' OF Sales ON REPORT
BREAK ON REPORT;

TTITLE LEFT 'The Top 3 Branch Based On Sales In Year '&v_year''SKIP2
SELECT * FROM QUERY1;

CLEAR BREAKS
CLEAR COMPUTES
CLEAR COLUMNS
TTITLE OFF
```

Sample Output:

```
SQL> @C:\Users\sharr\OneDrive\Desktop\Q1.sql

Session altered.

This query will generate top 3 outlet based on sales with given input as following
Enter the year : 2019
old   8: AND EXTRACT(YEAR FROM OrderDate) = '&v_year'
new   8: AND EXTRACT(YEAR FROM OrderDate) = '2019'

View created.


The Top 3 Branch Based On Sales In Year 2019

Branch ID       Branch Name              Branch Email                       Branch Address                Total Sales
--------------- ------------------------ ---------------------------------- ------------------------------ ------------
B2              Danau Kota               sbritcher1@constantcontact.com     31 Melrose Court                $2,254.00
B7              Palm Mall                cbucklee6@goo.ne.jp                1794 Fulton Way                   $989.00
B4              Pavilion                 fbradberry3@ftc.gov                58 Buena Vista Lane               $945.00
                                                                                                           ------------
SUM                                                                                                          $4,188.00
```

### 4.1.2 Query 2: Top 5 Products Sold Within A Date Range

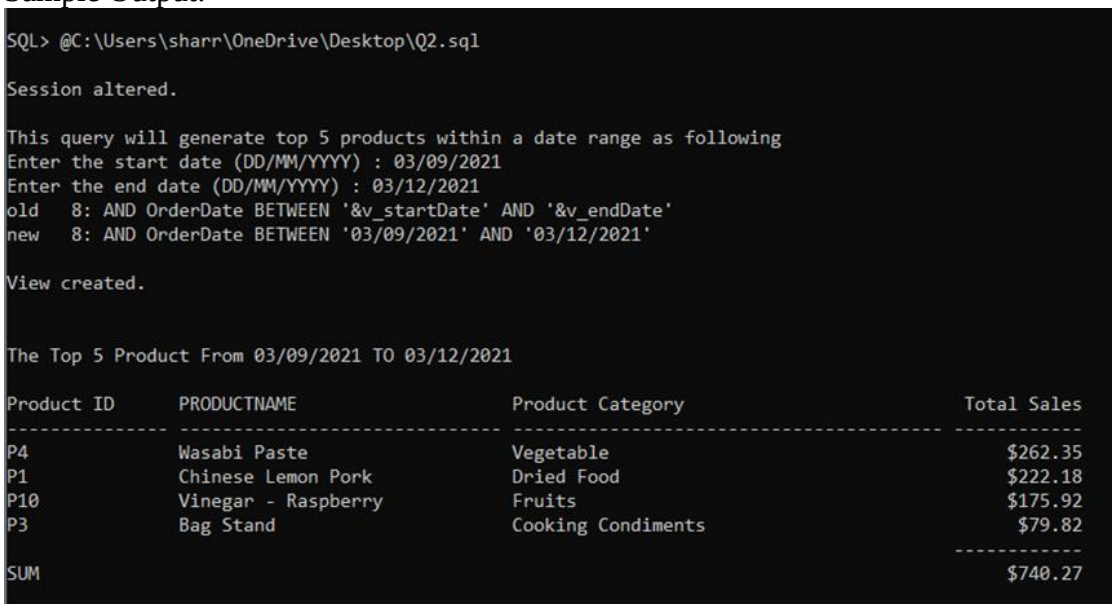
Purpose: To list out the top 5 products sold between StartDate and EndDate in all branches

SQL statement:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';
SET LINESIZE 180
SET PAGESIZE 100

COLUMN ProductID FORMAT A15 HEADING 'Product ID'
COLUMN ProdcutName FORMAT A25 HEADING 'Product Name'
COLUMN ProductCategory FORMAT A40 HEADING 'Product Category'
COLUMN Sales FORMAT $999,999.99 HEADING 'Total Sales'

PROMPT This query will generate top 5 products within a date range as
following
ACCEPT v_startDate DATE FORMAT 'DD/MM/YYYY' PROMPT 'Enter the start
date (DD/MM/YYYY) : '
ACCEPT v_endDate DATE FORMAT 'DD/MM/YYYY' PROMPT 'Enter the end date
(DD/MM/YYYY) : '
CREATE OR REPLACE VIEW QUERY2
AS SELECT *
FROM(
SELECT P.ProductID,ProductName,ProductCategory,SUM(Qty * UnitPrice) AS
Sales
FROM Product P,OrderDetails OD,Orders O
WHERE P.ProductID = OD.ProductID
AND O.OrderID = OD.OrderID
AND OrderDate BETWEEN '&v_startDate' AND '&v_endDate'
GROUP BY P.ProductID,ProductName,ProductCategory
ORDER BY Sales DESC)
WHERE ROWNUM < 5;
```

```
COMPUTE SUM LABEL 'SUM' OF Sales ON REPORT
BREAK ON REPORT;

TTITLE LEFT 'The Top 5 Product From '&v_startDate' TO
'&v_endDate''SKIP2
SELECT * FROM QUERY2;

CLEAR BREAKS
CLEAR COMPUTES
CLEAR COLUMNS
TTITLE OFF
```

Sample Output:



```
SQL> @C:\Users\sharr\OneDrive\Desktop\Q2.sql

Session altered.

This query will generate top 5 products within a date range as following
Enter the start date (DD/MM/YYYY) : 03/09/2021
Enter the end date (DD/MM/YYYY) : 03/12/2021
old   8: AND OrderDate BETWEEN '&v_startDate' AND '&v_endDate'
new   8: AND OrderDate BETWEEN '03/09/2021' AND '03/12/2021'

View created.


The Top 5 Product From 03/09/2021 TO 03/12/2021

Product ID      PRODUCTNAME                    Product Category                         Total Sales
--------------- ------------------------------ ---------------------------------------- -----------
P4              Wasabi Paste                   Vegetable                                   $262.35
P1              Chinese Lemon Pork             Dried Food                                  $222.18
P10             Vinegar - Raspberry            Fruits                                      $175.92
P3              Bag Stand                      Cooking Condiments                           $79.82
                                                                                        -----------
SUM                                                                                         $740.27
```

### 4.1.3 Procedure 1 :Top 3 Products Sold In A Branch Within A Date Range

Purpose: List out the top 3 products in a branch between the StartDate and EndDate

SQL statement:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';
SET SERVEROUTPUT ON
SET PAGESIZE 180
SET LINESIZE 180

--Extra efforts: Function
CREATE OR REPLACE FUNCTION getName (v_branchId IN VARCHAR)
RETURN VARCHAR IS
v_branchName VARCHAR(30);
BEGIN
SELECT BranchName INTO v_branchName
FROM Branch
WHERE BranchID = v_branchID;
RETURN v_branchName;
```

```
END;
/

CREATE OR REPLACE PROCEDURE prc_top3_prod_of_branch (v_branchID IN
VARCHAR, v_startDate IN VARCHAR, v_endDate IN VARCHAR) IS
v_prodID Product.productID%TYPE;
v_prodName Product.productname%TYPE;
v_unitPrice Product.unitprice%TYPE;
v_orderQty NUMBER(4);
v_totalSales NUMBER(9,2);
v_grandTotal NUMBER(9,2);

CURSOR prodCursor IS
SELECT *
FROM (SELECT P.productID, productname, unitprice, SUM(Qty) AS
TotalQuantity, SUM(unitprice * Qty) AS TotalSales
FROM Product P,OrderDetails OD,Orders O, Staff S,Branch B
WHERE P.ProductID = OD.ProductID
AND O.OrderID = OD.OrderID
AND O.StaffID = S.StaffID
AND S.BranchID = B.BranchID
AND orderDate BETWEEN v_startDate AND v_endDate
AND B.BranchID = v_branchId
GROUP BY P.productID, productName, unitPrice
ORDER BY TotalQuantity desc)
WHERE ROWNUM <= 3;
BEGIN

OPEN prodCursor;
 v_grandTotal := 0;
 DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
 DBMS_OUTPUT.PUT_LINE(CHR(10));
 DBMS_OUTPUT.PUT_LINE('Top 3 Product Sales In ' || getName(v_branchID)
||' from ' || v_startDate || ' to ' || v_endDate);
 DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
 DBMS_OUTPUT.PUT_LINE(RPAD('Product ID',18, ' ') || ' ' ||
 RPAD('Product Name',24, ' ') || ' ' ||
 RPAD('Price per Unit',20, ' ') || ' ' ||
 RPAD('Quantity Sold',24, ' ') || ' ' ||
 RPAD('Total Sales',14, ' '));
 DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
 LOOP
  FETCH prodCursor INTO v_prodID, v_prodName, v_unitPrice, v_orderQty,
v_totalSales;
  EXIT WHEN prodCursor%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE(RPAD(v_prodID,15, ' ') || ' ' ||
  RPAD(v_prodName,28, ' ') || ' ' ||
```

```
  RPAD(TO_CHAR(v_unitPrice,'$9,999.99'),22, ' ') || ' ' ||
  RPAD(v_orderQty,17, ' ') || ' ' ||
  RPAD(TO_CHAR(v_totalSales,'$9,999,999.99'),14, ' '));
  v_grandTotal := v_grandTotal + v_totalSales;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
 DBMS_OUTPUT.PUT_LINE(RPAD('Total amount of TOP 3 product sales: ',86,'
') || TO_CHAR(v_grandTotal,'$9,999,999.99'));
 DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
CLOSE prodCursor;
END;
/
--EXEC prc_top3_prod_of_branch ('B5','14/06/2020', '14/08/2022')
```

Sample Output:

```
SQL> EXEC prc_top10_prod_of_branch ('B5','14/06/2020', '14/08/2022')
================================================================================================================================


Top 3 Product Sales In Lalaport from 14/06/2020 to 14/08/2022
================================================================================================================================
Product ID         Product Name              Price per Unit       Quantity Sold           Total Sales
================================================================================================================================
P2             Garlic - Primerba, Paste         $17.63              13                    $229.19
P3             Bag Stand                         $6.14              9                      $55.26
P7             Syrup - Pancake                  $15.26              7                     $106.82
================================================================================================================================
Total amount of TOP 3 product sales:                                                      $391.27
================================================================================================================================

PL/SQL procedure successfully completed.
```

### 4.1.4 Procedure 2 : Sales and Demand Details of a Product in a Given Year

Purpose: List out the quantity sold and sales of a product, demand of the product (low /high) will be determined based on the quantity sold of the product in a year.

SQL statement:

```
SET SERVEROUTPUT ON
SET PAGESIZE 100
SET LINESIZE 120

CREATE OR REPLACE PROCEDURE prc_product_demand (v_prodID IN CHAR,
v_year IN VARCHAR) IS
v_prodName Product.ProductName%TYPE;
v_unitPrice Product.UnitPrice%TYPE;
v_qty NUMBER(4);
v_totalSales NUMBER(9,2);
v_prodStatus VARCHAR(30);

CURSOR prodCursor IS
SELECT *
FROM (SELECT ProductName, UnitPrice, SUM(Qty) AS TotalQuantity,
(UnitPrice * SUM(Qty)) AS TotalSales
```

```
FROM OrderDetails OD, Product P,Orders O
WHERE OD.productID = P.productID
AND OD.OrderID = O.OrderID
AND P.productID = v_prodID
AND EXTRACT(YEAR FROM OrderDate) = v_year
GROUP BY ProductName, UnitPrice);
BEGIN

OPEN prodCursor;
 LOOP
  FETCH prodCursor INTO v_prodName, v_unitPrice, v_qty, v_totalSales;
  EXIT WHEN prodCursor%NOTFOUND;
 END LOOP;
CLOSE prodCursor;

IF (v_qty <= 130) THEN
 v_prodStatus := 'Low Demand';
ELSE
 v_prodStatus := 'High Demand';
END IF;

DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
DBMS_OUTPUT.PUT_LINE(CHR(10));
DBMS_OUTPUT.PUT_LINE('Sales And Demand Details Of ' || v_prodID || ' In
Year ' || v_year);
DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
DBMS_OUTPUT.PUT_LINE(RPAD('Product Name',30, ' ') || ' ' ||
RPAD('Quantity Sold',20, ' ') || ' ' ||
RPAD('Price per Unit',20, ' ') || ' ' ||
RPAD('Demand of Product',20, ' ') || ' ' ||
RPAD('Total Sales',14, ' '));
DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
DBMS_OUTPUT.PUT_LINE(RPAD(v_prodName,30, ' ') || ' ' ||
RPAD(v_qty,16, ' ') || ' ' ||
RPAD(TO_CHAR(v_unitPrice,'$9,999.99'),24, ' ') || ' ' ||
RPAD(v_prodStatus,15, ' ') || ' ' ||
RPAD(TO_CHAR(v_totalSales,'$999,999.99'),14, ' '));
DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
END;
/
--EXEC prc_product_demand ('P7', '2022')
```

Sample Output:

```
SQL> EXEC prc_product_demand ('P7', '2022')
================================================================================================================

Sales And Demand Details Of P7 In Year 2022
================================================================================================================
Product Name                  Quantity Sold        Price per Unit       Demand of Product    Total Sales
================================================================================================================
Syrup - Pancake               43                   $15.26               Low Demand           $656.18
================================================================================================================

PL/SQL procedure successfully completed.
```

### 4.1.5 Trigger 1 : Manage Order Details Input

Purpose: To ensure the newly inputted quantity in order details is smaller or equal to the quantity in stock of the product. The quantity in stock of a product will be updated accordingly after insert or update or delete an order details. This trigger will be ran before inserting, updating or deleting a record in OrderDetails table.

SQL statement:

```
SET linesize 150
SET pagesize 120

ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';
SET SERVEROUTPUT ON

CREATE OR REPLACE TRIGGER trg_manage_order
BEFORE INSERT OR UPDATE OR DELETE ON OrderDetails
FOR EACH ROW
DECLARE
v_qtyInStock Product.QtyInStock%TYPE;
v_orderID Orders.OrderID%TYPE;
v_adjustedQty Product.QtyInStock%TYPE;
BEGIN
 CASE
 WHEN INSERTING THEN
    SELECT OrderID INTO v_orderID
    FROM Orders
    WHERE OrderID = :NEW.OrderID;

    SELECT QtyInStock INTO v_qtyInStock
    FROM Product
    WHERE ProductID = :NEW.ProductID;

  IF (:NEW.Qty <= v_qtyInStock) THEN
    UPDATE Product
    SET QtyInStock = QtyInStock - :NEW.Qty
    WHERE ProductID = :NEW.ProductID;
  ELSE
    RAISE_APPLICATION_ERROR(-20000,'Quantity In Stock not enough.');
  END IF;

 WHEN UPDATING THEN
```

```
    SELECT QtyInStock INTO v_qtyInStock
    FROM Product
    WHERE ProductID = :NEW.ProductID;

    SELECT OrderID INTO v_orderID FROM Orders
    WHERE OrderID = :NEW.OrderID;
    v_adjustedQty := :NEW.Qty - :OLD.Qty;

IF(:NEW.Qty > :OLD.Qty) THEN
    IF (v_adjustedQty <= v_qtyInStock) THEN
        UPDATE Product
        SET QtyInStock = QtyInStock - v_adjustedQty
        WHERE ProductID = :NEW.ProductID;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Quantity In Stock not enough.');
    END IF;
ELSE
    UPDATE Product
    SET QtyInStock = QtyInStock - v_adjustedQty
    WHERE ProductID = :NEW.ProductID;
END IF;

 WHEN DELETING THEN
    UPDATE Product
    SET QtyInStock = QtyInStock + :OLD.Qty
    WHERE ProductID = :OLD.ProductID;
 END CASE;
EXCEPTION

WHEN NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20000,'Order / Product not exist.');
END;
/
--Show the product's initial QtyInStock
select *
from product
where productID='P4';

--Invalid input
--Try to insert new order details(qty > qtyInStock)
--Show the raise error
insert into OrderDetails (OrderID, ProductID, Qty, SellingPrice,
Subtotal)
values ('OR5', 'P4', 8000, 167, 253);

--Valid input
--insert new order details(qty < qtyInStock)
```

```
insert into OrderDetails (OrderID, ProductID, Qty, SellingPrice,
Subtotal)
values ('OR5', 'P4', 300, 167, 253);

--Show the product's latest QtyInStock
select *
from product
where productID='P4';

--Update the order details inserted just now(qty < qtyInStock)
UPDATE OrderDetails
SET Qty = 400
WHERE ProductID = 'P4'
AND OrderID = 'OR5';

--Display the orderdetails inserted just now
select *
from orderdetails
where orderId = 'OR5'
and productId = 'P4';


--Show the product's latest qtyInstock(initial - ordered)
select *
from product
where productID='P4';

--Delete the order details inserted just now
DELETE FROM OrderDetails
WHERE ProductID = 'P4'
AND OrderID = 'OR5';

--The qtyInStock back to original value
select *
from product
where productID='P4';
```

Sample Output:

```
Trigger created.


PRODU PRODUCTNAME                     UNITPRICE QTYINSTOCK PRODUCTCATEGORY
----- ------------------------------ ---------- ---------- ------------------------
P4    Wasabi Paste                        17.49        820 Vegetable

insert into OrderDetails (OrderID, ProductID, Qty, SellingPrice, Subtotal)
            *
ERROR at line 1:
ORA-20000: Quantity In Stock not enough.
ORA-06512: at "SHARRENY2.TRG_MANAGE_ORDER", line 21
ORA-04088: error during execution of trigger 'SHARRENY2.TRG_MANAGE_ORDER'



1 row created.


PRODU PRODUCTNAME                     UNITPRICE QTYINSTOCK PRODUCTCATEGORY
----- ------------------------------ ---------- ---------- ------------------------
P4    Wasabi Paste                        17.49        520 Vegetable


1 row updated.


ORDER PRODU        QTY SELLINGPRICE   SUBTOTAL
----- ----- ---------- ------------ ----------
OR5   P4           400          167        253


PRODU PRODUCTNAME                     UNITPRICE QTYINSTOCK PRODUCTCATEGORY
----- ------------------------------ ---------- ---------- ------------------------
P4    Wasabi Paste                        17.49        420 Vegetable


1 row deleted.


PRODU PRODUCTNAME                     UNITPRICE QTYINSTOCK PRODUCTCATEGORY
----- ------------------------------ ---------- ---------- ------------------------
P4    Wasabi Paste                        17.49        820 Vegetable
```

### 4.1.6 Trigger 2 : Manage Staff Input

Purpose: To ensure the newly inputted staff is not having the same staff role with another 3 staffs in the same department(same branch), in valid format of staff ID, valid format of phone number, and older than 18 years old. No staff records can be

delete also.This trigger will be ran before inserting, updating or deleting a record in Staff table.

*Assumption*

*-In each branch, a department can only have 3 staffs with the same staff role*

*-Staff must be older than 18 years old*

*-Removal of staff records is not allowed*

SQL statement:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';
SET LINESIZE 180
SET PAGESIZE 100
--To check the format of staff id whether it is a 'ST' followed by 2
digits
CREATE OR REPLACE FUNCTION fun_checkID (id_in IN CHAR)
RETURN boolean IS
BEGIN
RETURN REGEXP_LIKE(id_in, 'ST\d{1,3}$');
END;
/

-- To check the staff age whether he/she is greater than 18 years old
CREATE OR REPLACE FUNCTION fun_checkAge (dob_in IN DATE)
RETURN boolean IS
BEGIN
RETURN extract(YEAR FROM (SYSDATE - dob_in) YEAR TO MONTH) > 18;
END;
/

--To check the format of the staff phone whether it is
--starts with a + symbol followed by 1 to 3 digits
--and followed by a space, 3 digits, a space, 3 digits, a space and 4
digits
CREATE OR REPLACE FUNCTION fun_checkPhone (phone_in IN VARCHAR2)
RETURN boolean IS
BEGIN
RETURN REGEXP_LIKE(phone_in, '[+]\d{1,3} \d{3} \d{3} \d{4}$');
END;
/

CREATE OR REPLACE TRIGGER trg_manage_staff
BEFORE INSERT OR UPDATE OR DELETE ON Staff
FOR EACH ROW

DECLARE
v_numStaff NUMBER(2);
```

```
v_maxStaff NUMBER(2);

BEGIN
 CASE
 WHEN INSERTING THEN
    IF (NOT(fun_checkID(:NEW.staffID))) THEN
        RAISE_APPLICATION_ERROR(-20000,'Invalid Staff ID');
    END IF;

    IF (NOT(fun_checkAge(:NEW.staffDob))) THEN
        RAISE_APPLICATION_ERROR(-20000,'Invalid Staff Age');
    END IF;

    IF (NOT(fun_checkPhone(:NEW.phone))) THEN
        RAISE_APPLICATION_ERROR(-20000,'Invalid Staff Phone Number');
    END IF;

    v_maxStaff := 3;
    SELECT countStaff INTO v_numStaff
    FROM (
            SELECT branchID, department, staffRole, COUNT(staffID) AS
countStaff
            FROM Staff
            WHERE branchID = :NEW.branchID
            AND department = :NEW.department
            AND staffRole = :NEW.staffRole
            GROUP BY branchID, department,staffRole);
    IF(v_numStaff >= v_maxStaff) THEN
        RAISE_APPLICATION_ERROR(-20000,'Number of staff reach
maximum.');
    END IF;

    WHEN UPDATING THEN
        IF (NOT(fun_checkID(:NEW.staffID))) THEN
            RAISE_APPLICATION_ERROR(-20000,'Invalid Staff ID');
        END IF;

        IF (NOT(fun_checkAge(:NEW.staffDob))) THEN
            RAISE_APPLICATION_ERROR(-20000,'Invalid Staff Age');
        END IF;

        IF (NOT(fun_checkPhone(:NEW.phone))) THEN
            RAISE_APPLICATION_ERROR(-20000,'Invalid Staff Phone
Number');
        END IF;
    WHEN DELETING THEN
        RAISE_APPLICATION_ERROR(-20000,'Cannot delete staff record.');
```

```
 END CASE;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        v_numStaff := 0;
END;
/
--
SELECT branchID, department, staffRole, COUNT(staffID) AS countStaff
FROM Staff
WHERE branchID = 'B5'
AND StaffRole = 'Baggers'
AND Department = 'Production'
GROUP BY branchID, department,staffRole;

--No.of staff reach maximum
INSERT INTO Staff(StaffID, Name, Gender, StaffDOB, Phone, StaffRole,
Salary, Address, RegDate, ExperienceInYear, Department, BranchID)
VALUES ('ST450', 'Tan', 'F', '21/02/1993', '+3 804 335 5222',
'Baggers', 3748, '165 Magdeline Drive', '28/04/2018', 8, 'Production',
'B5');

-- Invalid ID
INSERT INTO Staff
VALUES ('S2', 'Ferdinand Sturt', 'F', '20/04/1999', '+1 321 553 2269',
'Product Buyer', 4372, '0476 Southridge Court', '26/08/2015', 5, 'Human
Resources', 'B7');

--Invalid age
INSERT INTO  Staff
VALUES ('ST12', 'Ebeneser Janus', 'F', '21/10/2010', '+1 763 710 4612',
'Department Manager', 3622, '5429 Merchant Junction', '20/06/2019', 3,
'Marketing', 'B3');

--Invalid phone no
INSERT INTO  Staff
VALUES('ST6', 'Mathilda Marzella', 'M', '27/12/1999', '3 321 565 1915',
'Shipping and receiving clerks', 2350, '02227 Loomis Court',
'28/12/2020', 8, 'Human Resources', 'B9');

--Insert new valid staff
INSERT INTO Staff
VALUES('ST999', 'Yeoh', 'M', '30/11/2001', '+7 894 478 1553', 'Product
Buyer', 4894, '8485 Buell Parkway', '24/06/2019', 4, 'Human Resources',
'B9');

--Show new staff added
SELECT StaffID,StaffDOB,Phone,StaffRole,Department,BranchID
```

```
FROM Staff
WHERE StaffID = 'ST999';

--Update staff with invalid id
UPDATE Staff
SET StaffID = 'S999'
WHERE Name = 'Yeoh';

--Update staff with invalid age
UPDATE Staff
SET StaffDOB = '23/12/2010'
WHERE Name = 'Yeoh';

--Update staff with invalid phone
UPDATE Staff
SET Phone = '7 894 478 1553'
WHERE Name = 'Yeoh';

--Cannot delete staff record
DELETE FROM Staff
WHERE Name = 'Yeoh';
```

Sample Output:

```
Trigger created.


BRANC DEPARTMENT                STAFFROLE                      COUNTSTAFF
----- ------------------------- ------------------------------ ----------
B5    Production                Baggers                                 3

INSERT INTO Staff(StaffID, Name, Gender, StaffDOB, Phone, StaffRole, Salary, Add
            *
ERROR at line 1:
ORA-20000: Number of staff reach maximum.
ORA-06512: at "SHARRENY2.TRG_MANAGE_STAFF", line 30
ORA-04088: error during execution of trigger 'SHARRENY2.TRG_MANAGE_STAFF'


INSERT INTO Staff
            *
ERROR at line 1:
ORA-20000: Invalid Staff ID
ORA-06512: at "SHARRENY2.TRG_MANAGE_STAFF", line 9
ORA-04088: error during execution of trigger 'SHARRENY2.TRG_MANAGE_STAFF'


INSERT INTO  Staff
             *
ERROR at line 1:
ORA-20000: Invalid Staff Age
ORA-06512: at "SHARRENY2.TRG_MANAGE_STAFF", line 13
ORA-04088: error during execution of trigger 'SHARRENY2.TRG_MANAGE_STAFF'


INSERT INTO  Staff
             *
ERROR at line 1:
ORA-20000: Invalid Staff Phone Number
ORA-06512: at "SHARRENY2.TRG_MANAGE_STAFF", line 17
ORA-04088: error during execution of trigger 'SHARRENY2.TRG_MANAGE_STAFF'



1 row created.
```

```
STAFF STAFFDOB   PHONE            STAFFROLE                     DEPARTMENT                BRANC
----- ---------- ---------------- ----------------------------- ------------------------- -----
ST999 30/11/2001 +7 894 478 1553  Product Buyer                 Human Resources           B9

UPDATE Staff
       *
ERROR at line 1:
ORA-20000: Invalid Staff ID
ORA-06512: at "SHARRENY2.TRG_MANAGE_STAFF", line 35
ORA-04088: error during execution of trigger 'SHARRENY2.TRG_MANAGE_STAFF'


UPDATE Staff
       *
ERROR at line 1:
ORA-20000: Invalid Staff Age
ORA-06512: at "SHARRENY2.TRG_MANAGE_STAFF", line 39
ORA-04088: error during execution of trigger 'SHARRENY2.TRG_MANAGE_STAFF'


UPDATE Staff
       *
ERROR at line 1:
ORA-20000: Invalid Staff Phone Number
ORA-06512: at "SHARRENY2.TRG_MANAGE_STAFF", line 43
ORA-04088: error during execution of trigger 'SHARRENY2.TRG_MANAGE_STAFF'


DELETE FROM Staff
            *
ERROR at line 1:
ORA-20000: Cannot delete staff record.
ORA-06512: at "SHARRENY2.TRG_MANAGE_STAFF", line 46
ORA-04088: error during execution of trigger 'SHARRENY2.TRG_MANAGE_STAFF'
```

### 4.1.7 Report 1 : Orders Within a Date Range

Purpose: List out the orders and total sales of all orders within the date range.

SQL statement:

```
SET linesize 120
SET pagesize 100
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';
SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE prc_report1 (v_startDate IN DATE,v_endDate
IN DATE) IS
v_orderID Orders.orderID%TYPE;
v_orderDate Orders.orderDate%TYPE;
v_customerName Customer.custName%TYPE;
v_staffName Staff.name%TYPE;
v_subtotal NUMBER(11,2);
v_grandTotal NUMBER(11,2);
v_totalValue NUMBER(15,2);

CURSOR orderCursor IS
 SELECT orderID, orderDate, custName, name
 FROM Orders O, Customer C, Staff S
 WHERE O.custID = C.custID
 AND O.staffID = S.staffID
```

```
 AND orderDate BETWEEN v_startDate AND v_endDate
 ORDER BY orderDate;
CURSOR orderDetailCursor IS
 SELECT OD.productID, productName, unitPrice, qty, unitPrice * qty AS
subtotal
 FROM OrderDetails OD, Product P
 WHERE P.productID = OD.productID AND orderID = v_orderID;
 ordRec orderDetailCursor%ROWTYPE;
BEGIN
v_totalValue := 0;
OPEN orderCursor;
 DBMS_OUTPUT.PUT_LINE(CHR(10));
 DBMS_OUTPUT.PUT_LINE(LPAD('=',20, '=') || '=' || RPAD('Report of
Orders from ' || v_startDate || ' to ' || v_endDate , 71 ,'='));
 DBMS_OUTPUT.PUT_LINE(CHR(10));
 LOOP
  FETCH orderCursor INTO v_orderID, v_orderDate, v_customerName,
v_staffName;
  EXIT WHEN orderCursor%NOTFOUND;

  DBMS_OUTPUT.PUT_LINE(LPAD('=',92,'='));
  DBMS_OUTPUT.PUT_LINE(LPAD('*',1,' ') || LPAD('Order '|| v_orderID,
47 ,' '));
  DBMS_OUTPUT.PUT_LINE(LPAD('=',92,'=') || CHR(10));
  DBMS_OUTPUT.PUT_LINE('Order Date: '|| v_orderDate);
  DBMS_OUTPUT.PUT_LINE('Customer Name: '|| v_customerName);
  DBMS_OUTPUT.PUT_LINE('Staff Name: '|| v_staffName || CHR(10));
  DBMS_OUTPUT.PUT_LINE(LPAD('=',92,'='));
  DBMS_OUTPUT.PUT_LINE(RPAD('Product ID',15,' ')|| ' ' ||
  RPAD('Product Name',30,' ')|| ' ' ||
  RPAD('Unit Price',17,' ')|| ' ' ||
  RPAD('Quantity',15,' ')|| ' ' ||
  LPAD('Subtotal',10,' '));
  DBMS_OUTPUT.PUT_LINE(LPAD('=',92,'='));

  OPEN orderDetailCursor;
  v_grandTotal := 0;
   LOOP
    FETCH orderDetailCursor INTO ordRec;
    IF (orderDetailCursor%ROWCOUNT = 0) THEN
     DBMS_OUTPUT.PUT_LINE('No such order');
    END IF;
    EXIT WHEN orderDetailCursor%NOTFOUND;
    v_grandTotal := v_grandTotal + ordRec.subtotal;
    DBMS_OUTPUT.PUT_LINE(RPAD(ordRec.productID,13,' ')|| ' ' ||
    RPAD(ordRec.productName,29,' ') || ' ' ||
    RPAD(TO_CHAR(ordRec.unitPrice,'$9,999.99'),23,' ')|| ' ' ||
```

```
    RPAD(ordRec.qty,11,' ') || ' ' ||
    RPAD(TO_CHAR(ordRec.subtotal,'$999,999.99'),15,' '));
   END LOOP;

  v_totalValue := v_totalValue + v_grandTotal;
  DBMS_OUTPUT.PUT_LINE(LPAD('=',92,'='));
  DBMS_OUTPUT.PUT_LINE(RPAD('*',50, ' ') || LPAD('Grand Total: '||
TO_CHAR(v_grandTotal,'$999,999,999.99'),42,' '));
  DBMS_OUTPUT.PUT_LINE('No. of products: '||
orderDetailCursor%ROWCOUNT);
  DBMS_OUTPUT.PUT_LINE(CHR(10));
 CLOSE orderDetailCursor;
END LOOP;

DBMS_OUTPUT.PUT_LINE(LPAD(' ',38,'=') || RPAD('End of Report
',54,'='));

DBMS_OUTPUT.PUT_LINE('Total number of orders : ' ||
orderCursor%ROWCOUNT);
DBMS_OUTPUT.PUT_LINE('Total value of all orders in between ' ||
v_startDate || ' to ' || v_endDate || ': ' ||
TO_CHAR(v_totalValue,'$999,999,999,999.99'));
CLOSE orderCursor;
END;
/
--EXEC prc_report1('01/05/2021','01/08/2021')
```

Sample Output:

```
SQL> EXEC prc_report1('01/05/2021','01/08/2021')


====================Report of Orders from 01/05/2021 to 01/08/2021=========================


================================================================================================
*                                    Order OR53
================================================================================================

Order Date: 05/05/2021
Customer Name: Ruos
Staff Name: Calvin Duval

================================================================================================
Product ID      Product Name                  Unit Price        Quantity          Subtotal
================================================================================================
P1              Chinese Lemon Pork             $9.66              9                $86.94
P10             Vinegar - Raspberry           $14.66              1                $14.66
P2              Garlic - Primerba, Paste      $17.63              7               $123.41
================================================================================================
*                                                              Grand Total:        $225.01
No. of products: 3


================================================================================================
*                                    Order OR18
================================================================================================

Order Date: 25/07/2021
Customer Name: Ridger
Staff Name: Blanch Cottam

================================================================================================
Product ID      Product Name                  Unit Price        Quantity          Subtotal
================================================================================================
P2              Garlic - Primerba, Paste      $17.63              8               $141.04
P3              Bag Stand                      $6.14              12               $73.68
P4              Wasabi Paste                  $17.49              10              $174.90
================================================================================================
*                                                              Grand Total:        $389.62
No. of products: 3
```

```
================================================================================================
*                                        Order OR70
================================================================================================

Order Date: 26/07/2021
Customer Name: Beeton
Staff Name: Maryellen Turtle

================================================================================================
Product ID      Product Name                  Unit Price        Quantity          Subtotal
================================================================================================
P4              Wasabi Paste                    $17.49              10             $174.90
P6              Oats Large Flake                $10.13               6              $60.78
P7              Syrup - Pancake                 $15.26               2              $30.52
================================================================================================
*                                                              Grand Total:          $266.20
No. of products: 3


================================================================================================
*                                        Order OR19
================================================================================================

Order Date: 27/07/2021
Customer Name: Anthoine
Staff Name: Marti Cowill

================================================================================================
Product ID      Product Name                  Unit Price        Quantity          Subtotal
================================================================================================
P1              Chinese Lemon Pork              $9.66               3              $28.98
P6              Oats Large Flake                $10.13              2              $20.26
P9              Lemon Balm - Fresh              $3.78              10              $37.80
================================================================================================
*                                                              Grand Total:          $87.04
No. of products: 3


================================== End of Report ===============================================
Total number of orders : 4
Total value of all orders in between 01/05/2021 to 01/08/2021:             $967.87

PL/SQL procedure successfully completed.
```

### 4.1.8 Report 2 : Products Supplied by Each Supplier

Purpose: List out all products with supply details that is supplied by each supplier, the grand total of all products supplied by each supplier is calculated.

SQL statement:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';
SET SERVEROUTPUT ON
SET LINESIZE 150
SET PAGESIZE 120

CREATE INDEX Name
ON Product (productName);

CREATE OR REPLACE PROCEDURE prc_report2 IS
v_supplierID Supplier.supplierID%TYPE;
v_supplierName Supplier.supplierName%TYPE;
v_subtotal NUMBER(11,2);
```

```
v_grandTotal NUMBER(11,2);
v_totalvalue NUMBER(15,2);

CURSOR supplierCursor IS
 SELECT DISTINCT S.supplierID, supplierName
 FROM Supplier S, Supply SI
 WHERE S.supplierID = SI.supplierID
 ORDER BY LENGTH(S.supplierID), S.supplierID;

  CURSOR productDetailCursor IS
   SELECT DISTINCT P.productID, productName, productCategory,
supplyPrice, SUM(supplyQty) AS quantity, supplyPrice * SUM(supplyQty)
AS Subtotal
   FROM Product P, Supply SI, Supplier S
   WHERE S.supplierID = v_supplierID
   AND supplierName = v_supplierName
   AND P.productID = SI.productID
   AND S.supplierID = SI.supplierID
   GROUP BY P.productID, productName, supplyPrice, productCategory
   ORDER BY LENGTH(P.productID),P.productID, productName;
   prodRec productDetailCursor%ROWTYPE;
   BEGIN
   v_totalValue := 0;
    DBMS_OUTPUT.PUT_LINE(LPAD('=',27, '=') || '=' || RPAD('Report of
Products Supplied by Each Supplier', 72, '='));
    DBMS_OUTPUT.PUT_LINE(CHR(10));
    OPEN supplierCursor;
      LOOP
       FETCH supplierCursor INTO v_supplierID, v_supplierName;
       EXIT WHEN supplierCursor%NOTFOUND;
       DBMS_OUTPUT.PUT_LINE(LPAD('=',100, '='));
       DBMS_OUTPUT.PUT_LINE(CHR(10));
       DBMS_OUTPUT.PUT_LINE('Supplier ID: '|| v_supplierID);
       DBMS_OUTPUT.PUT_LINE('Supplier Name: '|| v_supplierName);
       DBMS_OUTPUT.PUT_LINE(CHR(10));
       DBMS_OUTPUT.PUT_LINE(LPAD('=',100,'='));
       DBMS_OUTPUT.PUT_LINE(RPAD('Product ID',10,' ') || ' ' ||
       RPAD('Product Name', 28, ' ') || ' ' ||
       RPAD('Product Category', 20, ' ') || ' ' ||
       RPAD('Quantity',9,' ')|| ' ' ||
       RPAD(' Buy Price',10,' ')|| ' ' ||
       LPAD('Subtotal',16,' '));
       DBMS_OUTPUT.PUT_LINE(LPAD('=',100,'='));

       OPEN productDetailCursor;
        v_grandTotal := 0;
        LOOP
```

```
          FETCH productDetailCursor INTO prodRec;
          IF (productDetailCursor%ROWCOUNT = 0) THEN
           DBMS_OUTPUT.PUT_LINE('No such product');
          END IF;
          EXIT WHEN productDetailCursor%NOTFOUND;
          v_grandTotal := v_grandTotal + prodRec.Subtotal;
          DBMS_OUTPUT.PUT_LINE(RPAD(prodRec.productID, 10, ' ') || ' '
||
          RPAD(prodRec.productName, 28, ' ') || ' ' ||
          RPAD(prodRec.productCategory, 20, ' ') || ' ' ||
          RPAD(prodRec.quantity, 9, ' ') || ' ' ||
          RPAD(TO_CHAR(prodRec.supplyPrice, '$9,999.99'), 12, ' ') || '
' ||
          RPAD(TO_CHAR(prodRec.Subtotal, '$9,999,999.99'), 20, ' '));
         END LOOP;

         v_totalValue := v_totalValue + v_grandTotal;
         DBMS_OUTPUT.PUT_LINE(LPAD('=',100, '='));
         DBMS_OUTPUT.PUT_LINE(RPAD('*', 50, ' ') || LPAD('Grand Total: '
||
         TO_CHAR(v_grandTotal, '$999,999,999.99'), 47, ' '));
         DBMS_OUTPUT.PUT_LINE('Total Record: ' ||
productDetailCursor%ROWCOUNT);

         DBMS_OUTPUT.PUT_LINE(CHR(10));
        CLOSE productDetailCursor;
       END LOOP;
    DBMS_OUTPUT.PUT_LINE(LPAD('=',43, '=') || '=' || RPAD('End of
Report', 56, '='));
    DBMS_OUTPUT.PUT_LINE(CHR(10));
    DBMS_OUTPUT.PUT_LINE('Total number of Supplier: ' ||
supplierCursor%ROWCOUNT);
    DBMS_OUTPUT.PUT_LINE('Total value of all product supplies: ' ||
TO_CHAR(v_totalValue, '$999,999,999,999.99'));
    CLOSE supplierCursor;
END;
/
DROP INDEX Name;
--EXEC prc_report2
```

Sample Output:

```
SQL> EXEC prc_report2
==========================Report of Products Supplied by Each Supplier==========================


================================================================================================


Supplier ID: SP1
Supplier Name: Dabjam


================================================================================================
Product ID Product Name                 Product Category     Quantity   Buy Price         Subtotal
================================================================================================
P1         Chinese Lemon Pork           Dried Food           342        $4,395.00    $1,503,090.00
P1         Chinese Lemon Pork           Dried Food           450        $4,982.00    $2,241,900.00
P1         Chinese Lemon Pork           Dried Food           659        $3,811.00    $2,511,449.00
P1         Chinese Lemon Pork           Dried Food           441        $4,125.00    $1,819,125.00
P1         Chinese Lemon Pork           Dried Food           344        $3,000.00    $1,032,000.00
P2         Garlic - Primerba, Paste     Fruits               654        $4,563.00    $2,984,202.00
P2         Garlic - Primerba, Paste     Fruits               697        $4,510.00    $3,143,470.00
P2         Garlic - Primerba, Paste     Fruits               465        $4,523.00    $2,103,195.00
P2         Garlic - Primerba, Paste     Fruits               625        $4,715.00    $2,946,875.00
P3         Bag Stand                    Cooking Condiments   438        $4,319.00    $1,891,722.00
P3         Bag Stand                    Cooking Condiments   327        $4,183.00    $1,367,841.00
P4         Wasabi Paste                 Vegetable            543        $4,198.00    $2,279,514.00
P4         Wasabi Paste                 Vegetable            498        $4,739.00    $2,360,022.00
P4         Wasabi Paste                 Vegetable            356        $2,087.00      $742,972.00
P5         Coffee - Decaffeinato Coffee Fresh Produce        324        $2,204.00      $714,096.00
P5         Coffee - Decaffeinato Coffee Fresh Produce        507        $2,350.00    $1,191,450.00
P5         Coffee - Decaffeinato Coffee Fresh Produce        466        $2,542.00    $1,184,572.00
P5         Coffee - Decaffeinato Coffee Fresh Produce        349        $4,909.00    $1,713,241.00
P6         Oats Large Flake             Fresh Produce        577        $4,863.00    $2,805,951.00
P6         Oats Large Flake             Fresh Produce        575        $2,428.00    $1,396,100.00
P7         Syrup - Pancake              Vegetable            536        $2,725.00    $1,460,600.00
P8         Chocolate Bar - Smarties     Vegetable            492        $3,438.00    $1,691,496.00
P8         Chocolate Bar - Smarties     Vegetable            522        $3,019.00    $1,575,918.00
P8         Chocolate Bar - Smarties     Vegetable            453        $4,731.00    $2,143,143.00
P9         Lemon Balm - Fresh           Cooking Condiments   319        $2,237.00      $713,603.00
P9         Lemon Balm - Fresh           Cooking Condiments   639        $3,383.00    $2,161,737.00
P9         Lemon Balm - Fresh           Cooking Condiments   548        $3,698.00    $2,026,504.00
P10        Vinegar - Raspberry          Fruits               544        $4,660.00    $2,535,040.00
P10        Vinegar - Raspberry          Fruits               473        $4,788.00    $2,264,724.00
P10        Vinegar - Raspberry          Fruits               411        $4,817.00    $1,979,787.00
================================================================================================
*                                                                    Grand Total:   $56,485,339.00
Total Record: 30


================================================================================================


Supplier ID: SP2
Supplier Name: Roodel
```

….SP2 to SP9

```
P10         Vinegar - Raspberry          Fruits               673         $3,891.00     $2,618,643.00
=============================================================================================================
*                                                                                  Grand Total:    $52,129,446.00
Total Record: 30


=============================================================================================================


Supplier ID: SP10
Supplier Name: Photofeed


=============================================================================================================
Product ID Product Name                 Product Category     Quantity   Buy Price          Subtotal
=============================================================================================================
P1         Chinese Lemon Pork           Dried Food           467        $3,375.00     $1,576,125.00
P2         Garlic - Primerba, Paste     Fruits               527        $2,106.00     $1,109,862.00
P2         Garlic - Primerba, Paste     Fruits               450        $4,563.00     $2,053,350.00
P2         Garlic - Primerba, Paste     Fruits               331        $3,592.00     $1,188,952.00
P2         Garlic - Primerba, Paste     Fruits               665        $3,203.00     $2,129,995.00
P3         Bag Stand                    Cooking Condiments   393        $3,183.00     $1,250,919.00
P3         Bag Stand                    Cooking Condiments   687        $2,145.00     $1,473,615.00
P3         Bag Stand                    Cooking Condiments   475        $4,169.00     $1,980,275.00
P4         Wasabi Paste                 Vegetable            338        $4,493.00     $1,518,634.00
P5         Coffee - Decaffeinato Coffee Fresh Produce        418        $3,845.00     $1,607,210.00
P6         Oats Large Flake             Fresh Produce        309        $2,231.00       $689,379.00
P6         Oats Large Flake             Fresh Produce        515        $3,547.00     $1,826,705.00
P6         Oats Large Flake             Fresh Produce        658        $2,586.00     $1,701,588.00
P7         Syrup - Pancake              Vegetable            500        $3,638.00     $1,819,000.00
P7         Syrup - Pancake              Vegetable            467        $3,529.00     $1,648,043.00
P7         Syrup - Pancake              Vegetable            310        $4,189.00     $1,298,590.00
P8         Chocolate Bar - Smarties     Vegetable            430        $4,297.00     $1,847,710.00
P8         Chocolate Bar - Smarties     Vegetable            332        $4,554.00     $1,511,928.00
P8         Chocolate Bar - Smarties     Vegetable            544        $2,382.00     $1,295,808.00
P8         Chocolate Bar - Smarties     Vegetable            354        $4,751.00     $1,681,854.00
P8         Chocolate Bar - Smarties     Vegetable            321        $3,245.00     $1,041,645.00
P9         Lemon Balm - Fresh           Cooking Condiments   666        $3,641.00     $2,424,906.00
P9         Lemon Balm - Fresh           Cooking Condiments   444        $2,861.00     $1,270,284.00
P9         Lemon Balm - Fresh           Cooking Condiments   652        $2,378.00     $1,550,456.00
P9         Lemon Balm - Fresh           Cooking Condiments   499        $4,787.00     $2,388,713.00
P10        Vinegar - Raspberry          Fruits               366        $2,503.00       $916,098.00
P10        Vinegar - Raspberry          Fruits               470        $2,473.00     $1,162,310.00
P10        Vinegar - Raspberry          Fruits               584        $4,972.00     $2,903,648.00
P10        Vinegar - Raspberry          Fruits               478        $3,629.00     $1,734,662.00
P10        Vinegar - Raspberry          Fruits               635        $3,230.00     $2,051,050.00
=============================================================================================================
*                                                                                  Grand Total:    $48,653,314.00
Total Record: 30


==========================================End of Report==========================================


Total number of Supplier: 10
Total value of all product supplies:      $524,724,208.00
```

## 4.2 Tong Zian Chuan

### 4.2.1 Query 1: Product below 50 of stocks.

Purpose: This query allows the market staff to get a list of products in low-stock quantity.

SQL statement:

```
PROMPT QUERY 1: Product below 50 of stocks.

SET linesize 200
SET pagesize 200
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';

COLUMN productID        FORMAT A20 Heading "Product ID";
COLUMN productName      FORMAT A30 Heading "Product Name";
COLUMN productCategory  FORMAT A20 Heading "Product Category";
COLUMN qtyInStock       FORMAT 9999 Heading "In Stock Quantity";

CREATE OR REPLACE VIEW qry_stock_below AS
SELECT p.productID,
       p.productName,
       p.productCategory,
       p.qtyInStock
FROM Product P
WHERE p.qtyInStock <=50
ORDER BY p.qtyInStock ASC;

TTITLE LEFT "Item that have stock below 50"

select * from qry_stock_below;
```

Sample Output:

```
Item that have stock below 20
Produ Product Name                 Product Category                                                                 In Stock Quantity
----- ---------------------------- -------------------------------------------------------------------------------- -----------------
P1    Chinese Lemon Pork           Dried Food                                                                                      12
P11   Beef ball                    Frozen Food                                                                                     12
P18   Frozen shrimp dumpling       Frozen Food                                                                                     13
P20   Almond milk                  Drinks                                                                                          14
P7    Syrup - Pancake              Vegetable                                                                                       14
P15   Orange juice                 Drinks                                                                                          16
P19   Strawberry                   Fruits                                                                                          22
P17   Sweet Corn                   Vegetable                                                                                       23
P6    Oats Large Flake             Fresh Produce                                                                                   28
P8    Chocolate Bar - Smarties     Vegetable                                                                                       33
P2    Garlic - Primerba, Paste     Fruits                                                                                          36
P16   Thai Milk Tea                Drinks                                                                                          45
```

### 4.2.2 Query 2: Staff registered year.

Purpose: This query allows the user to enter registration year to check which staff was register on the year.

SQL statement:

```
SET linesize 200
SET pagesize 200
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';

Accept v_regDate NUMBER FORMAT 9999  PROMPT "Please enter Register
Year: "

COLUMN BranchName FORMAT A20 HEADING "Branch Name";
COLUMN StaffID FORMAT A10 HEADING "Staff ID";
COLUMN Name FORMAT A20 HEADING "Staff Name";
COLUMN StaffRole FORMAT A30 HEADING "Staff Role";
COLUMN RegDate FORMAT A10 HEADING "Registration year";

SELECT BranchName, StaffID, Name, StaffRole, RegDate
FROM Branch b, Staff s
WHERE b.BranchID = s.BranchID AND extract(year from s.regDate) =
&v_regDate
ORDER BY s.StaffID ASC;

TTITLE LEFT "Staff that register in year " &v_regDate " and current
staff role is " &v_staffRole

select * from qry_register_year;
```

Sample Output:

```
Please enter Register Year: 2018
old   4: WHERE b.BranchID = s.BranchID AND extract(year from s.regDate) = &v_regDate
new   4: WHERE b.BranchID = s.BranchID AND extract(year from s.regDate) =       2018

View created.


Display year 2018 with the following staff
Branch Name          Staff ID   Staff Name           Staff Role                    Registration year
-------------------- ---------- -------------------- ----------------------------- -----------------
Puncak Jalil         ST1        Web Seys             Stock clerks                  28-04-2018
Serdang Raya         ST19       Patience Regenhardt  Cashiers                      11-08-2018
Puncak Jalil         ST22       Sella Ponder         Stock clerks                  12-12-2018
Tali Air             ST23       Danita Benne         Shipping and receiving clerks 17-07-2018
Pavilion             ST26       Massimo Godridge     Baggers                       11-11-2018
Amerin Mall          ST34       Yuma Meharry         Stock clerks                  25-06-2018
Pavilion             ST42       Bernardine Bhar      Store manager                 12-11-2018
Taming Jaya          ST48       Lisette Deddum       Shipping and receiving clerks 26-07-2018
Danau Kota           ST50       Feodora Meysham      Department Manager            15-11-2018
Danau Kota           ST51       Forrest Wildber      Baggers                       09-12-2018
Puncak Jalil         ST52       Dynah McElrath       Store manager                 14-10-2018
Amerin Mall          ST55       Fae Loverock         Stock clerks                  26-04-2018
Amerin Mall          ST63       Skyler Whyman        Baggers                       07-12-2018
Palm Mall            ST70       Reamonn McPartling   Cashiers                      23-12-2018
Palm Mall            ST72       Galen Carnew         Department Manager            19-08-2018
Serdang Raya         ST8        Trude Hinzer         Store manager                 04-09-2018
Tali Air             ST91       Tymothy Adamovicz    Baggers                       03-10-2018
East Coast           ST93       Orin Stokes          Product Buyer                 09-07-2018

18 rows selected.
```

### 4.2.3 Procedure 1: Customer refund

Purpose: This procedure will be called when a customer requests for order refunding. However an order that exceeds 3 days or a total amount greater than 300 is not allowed to be refunded.

SQL statement:

```
CREATE OR REPLACE PROCEDURE prc_refund(v_orderID IN CHAR) AS
  v_orderDate Orders.OrderDate%TYPE;
  v_totalAmount NUMBER(7,2);
  v_duration NUMBER(5);

BEGIN
  SELECT OrderDate INTO v_orderDate
  FROM Orders
  WHERE OrderID = v_orderID;

  SELECT SUM(UnitPrice * Qty) INTO v_totalAmount
  FROM Product P, OrderDetails OD
  WHERE P.ProductID = OD.ProductID AND OrderID = v_orderID
  GROUP BY OrderID;

  v_duration := ROUND(SYSDATE-v_OrderDate,0);

  IF (v_duration < 4) THEN
    IF(v_totalAmount <= 300) THEN
      DELETE FROM OrderDetails
      WHERE OrderID = v_orderID;
      DELETE FROM Orders

      WHERE OrderID = v_orderID;
      DBMS_OUTPUT.PUT_LINE('Refund Successfully.');
      DBMS_OUTPUT.PUT_LINE('Refund Amount : $' || v_totalAmount);
    ELSE
   DBMS_OUTPUT.PUT_LINE('ACTION FAILED. Total amount is greater than
$300');
       DBMS_OUTPUT.PUT_LINE('Please get the supervisor to handle the
situation.');
          DBMS_OUTPUT.PUT_LINE('Total amount : $' || v_totalAmount);
    END IF;
  ELSE
          DBMS_OUTPUT.PUT_LINE('ACTION FAILED. Order over 3 days.');
   DBMS_OUTPUT.PUT_LINE('Duration : '|| v_duration || 'day(s) ago');
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('The action is failed
Order ID entered is not exist');
END;
/

--failed, day more than 3 days
EXEC prc_refund('OR27')

--the total amount is more than 300
EXEC prc_refund('OR902')

--refundable
EXEC prc_refund('OR904')
```

Sample Output:

```
SQL> EXEC prc_refund('OR27')
ACTION FAILED. Order over 3 days.
Duration : 1379day(s) ago

PL/SQL procedure successfully completed.
```

Action failed, because the order day more than 3 days.

```
SQL> EXEC prc_refund('OR902')
ACTION FAILED. Total amount is greater than $300
Please get the supervisor to handle the situation.
Total amount : $756

PL/SQL procedure successfully completed.
```

Action failed because the total amount have greater than $300

```
SQL> EXEC prc_refund('OR904')
Refund Successfully.
Refund Amount : $146.6

PL/SQL procedure successfully completed.
```

Successfully refund

### 4.2.3 Procedure 2: Check in stock

Purpose: This procedure allows staff to view the specific month supply in stock after entering the year and month.

SQL statement:

```
CREATE OR REPLACE PROCEDURE
prc_supply_transaction_momth(supplyYear IN NUMBER, supplyMonth IN
NUMBER) IS
v_quantity NUMBER;
v_total NUMBER(11,2);
v_prodName Product.productName%TYPE;
v_supplyDate Supply.supplyDate%TYPE;
sumValue NUMBER(11,2);

CURSOR supply_Date_Cursor IS
  SELECT DISTINCT supplyDate
  FROM Supply
  WHERE EXTRACT(YEAR FROM supplyDate) = supplyYear AND
EXTRACT(MONTH FROM supplyDate) = supplyMonth ORDER BY supplyDate;

CURSOR productCursor IS
  SELECT P.ProductName, S.SupplyQty, SUM(P.unitPrice * S.SupplyQty)
AS SumValue
  FROM Supply S, Product P
  WHERE S.productID = P.productID AND supplyDate = v_supplyDate
  GROUP BY P.productName, S.SupplyQty;

prodRec productCursor%ROWTYPE;

BEGIN
OPEN supply_Date_Cursor;
LOOP
  FETCH supply_Date_Cursor INTO v_supplyDate;
  IF(supply_Date_Cursor%ROWCOUNT = 0) THEN
    DBMS_OUTPUT.PUT_LINE('No Record');
  END IF;
EXIT WHEN supply_Date_Cursor%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE(CHR(10));
    DBMS_OUTPUT.PUT_LINE('Transaction on supply Date: ' ||
v_supplyDate);
    DBMS_OUTPUT.PUT_LINE(RPAD('Product Name',40, ' ') || '' ||
RPAD('Total Quantity',40, ' ') || ' ' ||
  RPAD('Total Price', 5,' '));
    DBMS_OUTPUT.PUT_LINE(LPAD('=',100, '='));

OPEN productCursor;
  v_total := 0;
LOOP
  FETCH productCursor INTO prodRec;
  IF(productCursor%ROWCOUNT = 0) THEN
    DBMS_OUTPUT.PUT_LINE('No Record');
END IF;
EXIT WHEN productCursor%NOTFOUND;
  v_total := v_total + prodRec.SumValue;
    DBMS_OUTPUT.PUT_LINE(RPAD(prodRec.ProductName,40, ' ') || ' '
||
```

```
                            RPAD(prodRec.SupplyQty,34, ' ') || ' ' ||


              RPAD(TO_CHAR(prodRec.SumValue,'$999,999.99'),24, ' '));
END LOOP;

    DBMS_OUTPUT.PUT_LINE(LPAD('=',100, '='));
    DBMS_OUTPUT.PUT_LINE(CHR(10));
    DBMS_OUTPUT.PUT_LINE('Total Transaction : ' ||
TO_CHAR(v_total, '$999,999,999.99'));
CLOSE productCursor;
END LOOP;

EXCEPTION
WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No record found...');

CLOSE supply_Date_Cursor;
END;
/

EXEC prc_supply_transaction_momth(2022, 12);
```

Sample Output:

```
SQL> EXEC prc_supply_transaction_momth(2022, 1);


Transaction on supply Date: 10-01-2022
Product Name                              Total Quantity                                  Total
====================================================================================================
Syrup - Pancake                            431                                          $6,577.06
====================================================================================================


Total Transaction :        $6,577.06


Transaction on supply Date: 29-01-2022
Product Name                              Total Quantity                                  Total
====================================================================================================
Oats Large Flake                           515                                          $5,216.95
====================================================================================================


Total Transaction :        $5,216.95
```

### 4.2.5 Trigger 1: Validate the salary range

Purpose: Restrict users from inserting new employee information if the salary does not fit within the salary range

SQL statement:

```
CREATE OR REPLACE TRIGGER trg_check_staff_salary
BEFORE INSERT ON Staff
FOR EACH ROW
BEGIN
IF (:NEW.staffRole = 'security' AND :NEW.Salary
BETWEEN 2500 AND 3000) THEN
:NEW.staffRole := :NEW.Salary;

ELSIF (:NEW.staffRole = 'cashier' AND :NEW.Salary
BETWEEN 4500 AND 5000) THEN
:NEW.staffRole := :NEW.Salary;

ELSIF (:NEW.staffRole = 'customer service' AND :NEW.Salary
BETWEEN 3000 AND 3500) THEN
:NEW.staffRole := :NEW.Salary;

ELSIF (:NEW.staffRole = 'promoter' AND :NEW.Salary
BETWEEN 3000 AND 3800) THEN
:NEW.staffRole := :NEW.Salary;

ELSIF (:NEW.staffRole = 'stocker' AND :NEW.Salary
BETWEEN 4000 AND 4800) THEN
:NEW.staffRole := :NEW.Salary;

ELSIF (:NEW.staffRole = 'logistic' AND :NEW.Salary
BETWEEN 5000 AND 6000) THEN
:NEW.staffRole := :NEW.Salary;

ELSIF (:NEW.staffRole = 'fishmonger' AND :NEW.Salary
BETWEEN 4700 AND 5200) THEN
:NEW.staffRole := :NEW.Salary;

ELSIF (:NEW.staffRole = 'butcher' AND :NEW.Salary
BETWEEN 4800 AND 5200) THEN
:NEW.staffRole := :NEW.Salary;

ELSE
RAISE_APPLICATION_ERROR(-20000,'The salary is not in the salary
range or no such staff role');
END IF;
END;
/

--can be insert
INSERT INTO Staff VALUES ('ST999', 'Nils Bubbear', 'M',
'24/04/1961', '841-264-2138', 'stocker', 4200, '12 Alor Setar',
'22/12/2022', '5', 'Research and Development', 'B3');
DELETE FROM STAFF WHERE staff_ID IN ('S998');

--cannot insert because the salary is not in the salary range of
Stocker(4000-4800)
```

```
INSERT INTO Staff VALUES ('ST999', 'Nils Bubbear', 'M',
'24/04/1961', '841-264-2138', 'stocker', 3700, '12 Alor Setar',
'22/12/2022', '5', 'Research and Development', 'B3');
```

Sample output:

```
SQL> INSERT INTO Staff VALUES ('ST999', 'Nils Bubbear', 'M', '24/04/1961', '841-264-21
INSERT INTO Staff VALUES ('ST999', 'Nils Bubbear', 'M', '24/04/1961', '841-264-2138',
            *
ERROR at line 1:
ORA-20000: The salary is not in the salary range or no such staff role
ORA-06512: at "ZC.TRG_CHECK_STAFF_SALARY", line 35
ORA-04088: error during execution of trigger 'ZC.TRG_CHECK_STAFF_SALARY'
```

cannot insert because the salary is not in the salary range of Stocker (4000-4800)

```
SQL> INSERT INTO Staff VALUES ('ST999', 'Nils Bubbear', 'M', '24/04/1961', '841-264-2138', 'stocker', 4200, '12 Alor Setar', '22/12/2022', '5', 'Research and Development', 'B3');

1 row created.
```

can be insert since the staff salary is within its role's salary range

### 4.2.6 Trigger 2: Check modification on supplier

Purpose: To check who had done the insert, modify and delete on the table of Supplier

SQL statement:

```
SET PAGESIZE 200
SET LINESIZE 220
SET SERVEROUTPUT ON

DROP TABLE InsertSupplierRec;
DROP TABLE UpdateSupplierRec;
DROP TABLE DeleteSupplierRec;

CREATE TABLE InsertSupplierRec (
UserID VARCHAR(6),
EditDate DATE,
Time CHAR(8),
supplierID CHAR(11),
supplierName VARCHAR(25),
supplierPhone VARCHAR(15),
supplierEmail VARCHAR(30)
);

CREATE TABLE UpdateSupplierRec (
UserID VARCHAR(6),
EditDate DATE,
Time CHAR(8),
supplier_ID CHAR(11),
oldname VARCHAR(25),
newname VARCHAR(25),
oldphone VARCHAR(15),
newphone VARCHAR(15),
oldemail VARCHAR(30),
newemail VARCHAR(30)
);

CREATE TABLE DeleteSupplierRec (
UserID VARCHAR(6),
EditDate DATE,
Time CHAR(8),
supplierID CHAR(11),
supplierName VARCHAR(25),
supplierPhone VARCHAR(15),
supplierEmail VARCHAR(30)
);

CREATE OR REPLACE TRIGGER trg_perform_check
AFTER INSERT OR UPDATE OR DELETE ON Supplier
FOR EACH ROW
BEGIN
CASE
WHEN INSERTING THEN
INSERT INTO InsertSupplierRec
VALUES(USER, SYSDATE, TO_CHAR(SYSDATE,
'HH24:MI:SS'), :NEW.supplierID, :NEW.supplierName, :NEW.supplierPho
ne, :NEW.supplierEmail);
WHEN UPDATING THEN
INSERT INTO UpdateSupplierRec
```

```
VALUES(USER, SYSDATE, TO_CHAR(SYSDATE,
'HH24:MI:SS'), :OLD.supplierID, :OLD.supplierName, :NEW.supplierNam
e, :OLD.supplierPhone, :NEW.supplierPhone, :OLD.supplierEmail, :NEW
.supplierEmail);
WHEN DELETING THEN
INSERT INTO DeleteSupplierRec
VALUES(USER, SYSDATE, TO_CHAR(SYSDATE,
'HH24:MI:SS'), :OLD.supplierID, :OLD.supplierName, :OLD.supplierPho
ne, :OLD.supplierEmail);
END CASE;
END;
/

INSERT INTO Supplier VALUES ('SP50', 'zc', '1234567890',
'stanford12@gmail.com');

UPDATE Supplier SET supplierName = 'Ada Wong', supplierPhone =
'987654321' WHERE supplierID = 'SP50';

DELETE FROM Supplier WHERE supplierID = 'SP50';

SELECT * FROM InsertSupplierRec;
SELECT * FROM UpdateSupplierRec;
SELECT * FROM DeleteSupplierRec;
```

Sample output:

```
SQL> INSERT INTO Supplier VALUES ('SP50', 'zc', '1234567890', 'stanford12@gmail.com');

1 row created.

SQL> UPDATE Supplier
  2  SET supplierName = 'Ada Wong', supplierPhone = '987654321'
  3  WHERE supplierID = 'SP50';

1 row updated.

SQL> DELETE FROM Supplier WHERE supplierID = 'SP50';

1 row deleted.

SQL> SELECT * FROM InsertSupplierRec;

USERID EDITDATE   TIME     SUPPLIERID  SUPPLIERNAME             SUPPLIERPHONE   SUPPLIEREMAIL
------ ---------- -------- ----------- ------------------------ --------------- ------------------------------
ZC     26/12/2022 02:56:14 SP99        ZC                       1234567890      stanford12@gmail.com
ZC     26/12/2022 02:58:57 SP50        zc                       1234567890      stanford12@gmail.com

SQL> SELECT * FROM UpdateSupplierRec;

USERID EDITDATE   TIME     SUPPLIER_ID OLDNAME                  NEWNAME                  OLDPHONE        NEWPHONE        OLDEMAIL                       NEWEMAIL
------ ---------- -------- ----------- ------------------------ ------------------------ --------------- --------------- ------------------------------ ------------------------------
ZC     26/12/2022 02:56:52 SP99        ZC                       sb                       1234567890      987654321       stanford12@gmail.com           stanford12@gmail.com
ZC     26/12/2022 02:59:07 SP50        zc                       Ada Wong                 1234567890      987654321       stanford12@gmail.com           stanford12@gmail.com

SQL> SELECT * FROM DeleteSupplierRec;

USERID EDITDATE   TIME     SUPPLIERID  SUPPLIERNAME             SUPPLIERPHONE   SUPPLIEREMAIL
------ ---------- -------- ----------- ------------------------ --------------- ------------------------------
ZC     26/12/2022 02:56:06 SP99        Dejavu                   +676767676767   dejavu69@gmail.com
ZC     26/12/2022 02:57:00 SP99        sb                       987654321       stanford12@gmail.com
```

### 4.2.7 Report 1: List the total profit of the each branch

Purpose: List the total profit at each branch and sum the total profit of every branch during a period of time.

SQL statement:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';
SET SERVEROUTPUT ON
SET PAGESIZE 100
SET LINESIZE 120

CREATE OR REPLACE PROCEDURE prc_show_branch_sales
(v_startDate IN VARCHAR, v_endDate IN VARCHAR) IS

v_branchname Branch.branchname%TYPE;
v_grandTotal NUMBER(11,2);
v_totalValue NUMBER(15,2);

CURSOR branchCursor IS
SELECT BranchName
FROM Branch;

CURSOR itemCursor IS
SELECT * FROM (SELECT DISTINCT P.productID, productName,
unitprice, SUM(Qty) AS totalQty,
(SUM(Qty) * unitPrice) AS totalProfit
FROM Branch B, Staff S, Orders O, OrderDetails OD,
Product P
WHERE B.branchID = S.branchID AND S.staffID = O.staffID
AND O.orderID = OD.orderID AND OD.productID =
P.productID AND orderDate BETWEEN v_startDate AND v_endDate
AND branchname = v_branchname
GROUP BY P.productID, productname, unitprice
ORDER BY P.productID);

itemRec itemCursor%ROWTYPE;

BEGIN
v_totalValue := 0;
OPEN branchCursor;
LOOP
FETCH branchCursor INTO v_branchName;
EXIT WHEN branchCursor%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
DBMS_OUTPUT.PUT_LINE(CHR(10));
DBMS_OUTPUT.PUT_LINE('Branch Name: ' || v_branchName);
DBMS_OUTPUT.PUT_LINE(CHR(10));
DBMS_OUTPUT.PUT_LINE('Total profit in ' || v_branchName || '
during '|| v_startDate || ' to ' || v_endDate);
DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
DBMS_OUTPUT.PUT_LINE(RPAD('Product ID',13, ' ') || ' ' ||



                            RPAD('Product Name',30, ' ') || ' ' ||
```

```
                                    RPAD('Quantity',9, ' ') || ' ' ||


                                  RPAD('Unit Price',14, ' ') || ' ' ||


                                       RPAD('Total Profit',14, ' '));
DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
OPEN itemCursor;
v_grandTotal := 0;
LOOP
FETCH itemCursor INTO itemRec;
IF (itemCursor%ROWCOUNT = 0) THEN
DBMS_OUTPUT.PUT_LINE('No profit in ' || v_branchName || ' during
this period');
END IF;
EXIT WHEN itemCursor%NOTFOUND;
v_grandTotal := v_grandTotal + itemRec.totalProfit;
DBMS_OUTPUT.PUT_LINE(RPAD(itemRec.productID,13, ' ') || ' '||
                     RPAD(itemRec.productname,30, ' ') || ' ' ||
                     RPAD(itemRec.totalQty,9, ' ') || ' ' ||
                     RPAD(itemRec.unitprice,12, ' ') || ' ' ||

RPAD(TO_CHAR(itemRec.totalProfit,'$999,999.99'),14, ' '));
END LOOP;
CLOSE itemCursor;
v_totalValue := v_totalValue + v_grandTotal;
DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
DBMS_OUTPUT.PUT_LINE(RPAD('*', 26, ' ') || LPAD('GrandTotal: ' ||
TO_CHAR(v_grandTotal,'$9,999,999.99'),54, ' '));
DBMS_OUTPUT.PUT_LINE(CHR(10));
DBMS_OUTPUT.PUT_LINE(LPAD('=',50, '=') || '=' || RPAD('End Of
Report',69,'='));
DBMS_OUTPUT.PUT_LINE(CHR(10));
END LOOP;
DBMS_OUTPUT.PUT_LINE('Total number of branch            : '||
branchCursor%ROWCOUNT);
DBMS_OUTPUT.PUT_LINE('Total amount of all product profit: '||
TO_CHAR(v_totalValue,'$999,999.99'));
CLOSE branchCursor;
END;
/

EXEC prc_show_branch_sales ('01/01/2021', '31/12/2021')

EXEC prc_show_branch_sales ('01/01/2021', '31/12/2021')
```

Sample output:

```
SQL> EXEC prc_show_branch_sales ('01/01/2021', '31/12/2021')
================================================================================================================================


Branch Name: Serdang Raya


Total profit in Serdang Raya
during 01/01/2021 to 31/12/2021
================================================================================================================================
Product ID    Product Name                  Quantity  Unit Price     Total Profit
================================================================================================================================
P1            Chinese Lemon Pork            23        9.66             $222.18
P10           Vinegar - Raspberry           12        14.66            $175.92
P2            Garlic - Primerba, Paste      4         17.63             $70.52
P3            Bag Stand                     8         6.14              $49.12
P9            Lemon Balm - Fresh            10        3.78              $37.80
================================================================================================================================
*                                                     GrandTotal:       $555.54


================================================End Of Report=================================================
```

### 4.2.8 Report 2: Track customers purchasing

Purpose: List the customers who had bought before a certain product during a period of time at each branch and compute the grand total of each branch and total sales of the product.

SQL statement:

```
CREATE OR REPLACE PROCEDURE prc_custBuy
(v_prodID IN VARCHAR, v_startDate IN VARCHAR, v_endDate IN
VARCHAR) IS
v_branchName Branch.branchname%TYPE;
v_grandTotal NUMBER(11,2);
v_totalValue NUMBER(15,2);

CURSOR branchCursor IS
SELECT branchname
FROM Branch;

CURSOR prodCursor IS
SELECT P.productID, Qty, orderDate,
C.custID, custname, branchname, (Qty * unitprice) AS totalSales
FROM Product P, OrderDetails OD, Orders O, Customer
C, Staff S, Branch B
WHERE P.productID = OD.productID AND
      OD.orderID = O.orderID AND
      O.custID = C.custID AND
      O.staffID = S.staffID AND
      S.branchID = B.branchID AND
      P.productID = v_prodID AND
      B.branchname = v_branchName AND
      orderDate BETWEEN v_startDate AND v_endDate
ORDER BY orderDate;

prodRec prodCursor%ROWTYPE;

BEGIN
v_totalValue := 0;
OPEN branchCursor;
LOOP
FETCH branchCursor INTO v_branchName;
EXIT WHEN branchCursor%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
DBMS_OUTPUT.PUT_LINE(CHR(10));
DBMS_OUTPUT.PUT_LINE('Branch Name: ' || v_branchName);
DBMS_OUTPUT.PUT_LINE(CHR(10));
DBMS_OUTPUT.PUT_LINE('Customer Purchase product ' || v_prodID || '
during '|| v_startDate || ' to ' || v_endDate);
DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
DBMS_OUTPUT.PUT_LINE(RPAD('Customer ID',13, ' ') || ' ' ||
                     RPAD('Customer Name',25, ' ') || ' ' ||
                     RPAD('Quantity',9, ' ') || ' ' ||
                     RPAD('Date',14, ' ') || ' ' ||
                     RPAD('Total Sales',14, ' '));
DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));

OPEN prodCursor;

v_grandTotal := 0;
```

```
LOOP
FETCH prodCursor INTO prodRec;
  IF (prodCursor%ROWCOUNT = 0) THEN
    DBMS_OUTPUT.PUT_LINE('No customer purchase product ' ||
v_prodID || ' in ' || v_branchName || ' during this period');
  END IF;
EXIT WHEN prodCursor%NOTFOUND;

v_grandTotal := v_grandTotal + prodRec.totalSales;

    DBMS_OUTPUT.PUT_LINE(RPAD(prodRec.custID,13, ' ') || ' ' ||
                    RPAD(prodRec.custname,25, ' ') || ' ' ||
                    RPAD(prodRec.Qty,9, ' ') || ' ' ||
                    RPAD(prodRec.orderDate,12, ' ') || ' ' ||

RPAD(TO_CHAR(prodRec.totalSales,'$999,999.99'),14, ' '));
END LOOP;

v_totalValue := v_totalValue + v_grandTotal;

    DBMS_OUTPUT.PUT_LINE(LPAD('=',120, '='));
    DBMS_OUTPUT.PUT_LINE(RPAD('*', 22, ' ') || LPAD('GrandTotal: '
|| TO_CHAR(v_grandTotal,'$9,999,999.99'),54, ' '));
    DBMS_OUTPUT.PUT_LINE('Total Customer: ' ||
prodCursor%ROWCOUNT);
    DBMS_OUTPUT.PUT_LINE(LPAD('=',50, '=') || '=' || RPAD('End Of
Report',69,'='));
    DBMS_OUTPUT.PUT_LINE(CHR(10));
CLOSE prodCursor;

END LOOP;

    DBMS_OUTPUT.PUT_LINE('Total number of branch: ' ||
branchCursor%ROWCOUNT);
    DBMS_OUTPUT.PUT_LINE('Total amount of all product sales: '||
TO_CHAR(v_totalValue,'$999,999,999,999.99'));
CLOSE branchCursor;
END;
/


EXEC prc_custBuy ('P1', '01/01/2015','23/12/2022')
EXEC prc_custBuy ('P1', '01/01/2022','23/12/2022')
```

Sample output:

```
Branch Name: Puncak Jalil


Customer Purchase product P1 during 01/01/2015 to 23/12/2022
=============================================================================================================================
Customer ID   Customer Name             Quantity  Date           Total Sales
=============================================================================================================================
CU2           Vedekhov                  6         12/10/2016         $57.96
CU7           Tuffell                   7         17/03/2017         $67.62
CU2           Vedekhov                  1         17/08/2019          $9.66
CU6           Hoyes                     6         26/04/2021         $57.96
CU10          Anthoine                  3         27/07/2021         $28.98
CU10          Anthoine                  7         04/04/2022         $67.62
=============================================================================================================================
*                                                 GrandTotal:        $289.80
Total Customer: 6
================================================End Of Report================================================


Total number of branch: 10
Total amount of all product sales:           $1,999.62
```

```
Branch Name: Palm Mall


Customer Purchase product P1 during 01/01/2022 to 23/12/2022
=============================================================================================================================
Customer ID   Customer Name             Quantity  Date           Total Sales
=============================================================================================================================
No customer purchase product P1 in Palm Mall during this period
=============================================================================================================================
*                                                 GrandTotal:           $.00
Total Customer: 0
================================================End Of Report================================================


=============================================================================================================================
```

- when no customers have make purchase in the month

# Chapter 5 Extra Effort Highlights

## 5.1 Tang Sharren

### 5.1.1 Views

I created a view in my query 1 and query 2 of the data in one or more tables in the database. View is created and used as a security mechanism by allowing the users to access the data through the view that satisfied the conditions of the SQL statement without granting the user permission to directly access the original base table

### 5.1.2 Column

Column can help us to rename the column name to avoid misunderstanding the user. Title will print a title on the top of the table to let the user know what they are viewing. I have used it for my query 1 and query 2.

### 5.1.3 Indexes

I have applied this index in my report. Indexes are used to retrieve data from the database more quickly. It can speed up the searching process by using the index. So, I am using an index for the product name then it can let the program reduce the execution time when searching for the data.

### 5.1.4 User Defined Functions

Procedure 1

User defined function in my procedure 1 is to return the branch name based on the branch id passed into the function.

```
CREATE OR REPLACE FUNCTION getName (v_branchId IN VARCHAR)
RETURN VARCHAR IS
v_branchName VARCHAR(30);
BEGIN
SELECT BranchName INTO v_branchName
FROM Branch
WHERE BranchID = v_branchID;
RETURN v_branchName;
END;
/
```

Trigger 2

User defined functions in my trigger 2 includes the following:

(i) To check the format of staff id whether it is a 'ST' followed by 2 digits, if the input id is in valid format it will return true else false

```
CREATE OR REPLACE FUNCTION fun_validID (id_in IN CHAR)
RETURN boolean IS
BEGIN
RETURN REGEXP_LIKE(id_in, 'ST\d{1,3}$');
```

```
END;
/
```

(ii) To check the staff age whether he/she is older than 18 years old, if age is older than 18 years old it will return true else false

```
CREATE OR REPLACE FUNCTION fun_validAge (dob_in IN DATE)
RETURN boolean IS
BEGIN
RETURN extract(YEAR FROM (SYSDATE - dob_in) YEAR TO MONTH) > 18;
END;
/
```

(iii)To check the format of the staff phone whether it is starts with a + symbol followed by 1 to 3 digits and followed by a space, 3 digits, a space, 3 digits, a space and 4 digits, if the input phone is in valid format it will return true, else false.

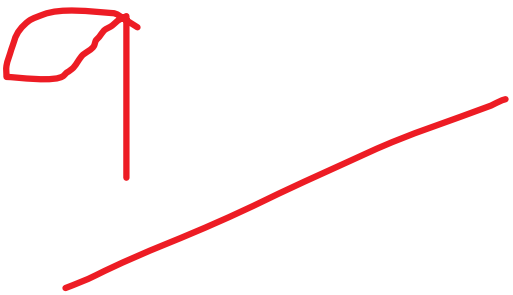
```
CREATE OR REPLACE FUNCTION fun_validPhone (phone_in IN VARCHAR2)

RETURN boolean IS
BEGIN
RETURN REGEXP_LIKE(phone_in, '[+]\d{1,3} \d{3} \d{3} \d{4}$');
END;
/
```

### 5.1.5 Prompt

I have applied prompt in query 1 and query 2. It is used to print text on the display for the user to read. It allows you to provide informative descriptions of what a script is about to do.

### 5.1.6 Accept prompt

I have applied accept prompt in query 1, query 2, procedure 1, procedure 2 and report 1, this is to get user input and store the user input in a variable to be used in my program.



## 5.2 Tong Zian Chuan

### 5.2.1 Views

In Query 1 where the view is applied. View is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

### 5.2.2 Case

In Trigger 2 where the case function is used. The CASE expression goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.

### 5.2.3 Prompt

Prompt used in Query 1 and 2. To print text on the display for the user to read, use the PROMPT command. You can use it to give detailed explanations of what a script is planning to do.

### 5.2.4 Accept prompt

In Query 2 the accept prompt is applied. The ACCEPT command is used to obtain input from the user. With it, you specify a user variable and text for a prompt.

### 5.2.5 Column

Column can assist us in renaming the column name so that it is more user-friendly for the user to see.

### 5.2.5 User defined exceptions

User Defined Exception is the exception that is explicitly defined and handled by the user. User Defined Exception has been implemented in my procedure and when the data is not recorded, it will prompt the exception.

### 5.2.6 Formatting Output

In query 1. Column is useful to define the format and heading of the column. The heading of the column can be changed to a more readable or understandable name instead of the variable name. The format can be used to standardize the format of the data to be displayed in each column. The application of the column in both query to format each of the columns to be displayed such as column of productID with format A20 and heading of “Product ID”. The column will make sure that the query is neatly presented with the format and heading.