

Faculty of Computing and Information Technology (FOCS)

Academic Year 2020/2021

**BACS3183**

**Advanced Database Management**

|  |  |  |
| --- | --- | --- |
| Programme | : | RDS3 |
| Tutorial Group | : | G2 |
| Date Submitted to Tutor | : | 01-09-2021 |

**Team Members:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Student Name** | **Student ID** | **Signature** |
| 1. | Tan Yi Hong | 20WMR08890 |  |
| 2. | Tan Teoh Xin Ee | 20WMR08887 |  |
| 3. | Tan Wei Siong | 20WMR08888 |  |
| 4. | Nigel Lee Jian Hsee | 20WMR08882 |  |

**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 | : |  |  |  |  |
| Name | : | Tan Yi Hong | Tan Teoh Xin Ee | Tan Wei Siong | Nigel Lee Jian Hsee |
| Date | : | 01-09-2021 | 01-09-2021 | 01-09-2021 | 01-09-2021 |

**Table of Content**

**Chapter 1 Background of the System 4**

**Chapter 2 Entity-Relationship Modeling 5**

**2.1 ERD 5**

**2.2 Assumptions and Business Rules 5**

**Chapter 3 Data Definition 8**

**Chapter 4 Queries, Procedures, Triggers and Reports 12**

**4.1 (Tan Yi Hong) 12**

**4.1.1 Query 1: Top pet type that received treatment in each branch (Strategic) 12**

**4.1.2 Query 2: Appointment made on times of a day in each branch for last year (Tactical) 13**

**4.1.3 Query 3: Year 2020 first half sales vs second half sales in each branch (Operational) 15**

**4.1.4 Procedure 1: Add Appointment Record 16**

**4.1.5 Procedure 2: Delete Appointment Record 18**

**4.1.6 Trigger 1: Validate the proper date time on insertion of appointment 18**

**4.1.7 Trigger 2: Monitor the deletion of appointment record 19**

**4.1.8 Report 1: Summary report of Specific veterinarian with all of his/her transactions done in a year 20**

**4.1.9 Report 2: Detail report of Customer list in specific state with appointment made in a year 25**

**4.1.10 Report 3: On demand report of Customer list who do transaction for less than three times in a year 30**

**4.2 (Tan Teoh Xin Ee) 32**

**4.2.1 Query 1: Top Medicine Used in each branch (Strategic) 32**

**4.2.2 Query 2: Top Veterinarian in each branch (Tactical) 33**

**4.2.3 Query 3: Late Supplier List (Operational) 34**

**4.2.4 Procedure 1: Add Medical Supply 35**

**4.2.5 Procedure 2: Delete Medical Supply 37**

**4.2.6 Trigger 1: Add Stock Quantity (Add after purchasing stock) 37**

**4.2.7 Trigger 2: Minus Stock Quantity (Minus after adding transaction detail) 38**

**4.2.8 Report 1: Detail report of Veterinarian Performance 38**

**4.2.9 Report 2: Summary report of Supplier Profile 40**

**4.2.10 Report 3: On demand report of Inventory value 43**

**4.3 (Tan Wei Siong) 45**

**4.3.1 Query 1: Top and Least Treatment Last Year for Each Branch (Strategic) 45**

**4.3.2 Query 2: Selected Pet Type Age Group and Respective Visit Quantity(Tactical) 46**

**4.3.3 Query 3: Gross Profit and Net Profit Previous Year(Operational) 48**

**4.3.4 Procedure 1: Pet Registration 49**

**4.3.5 Procedure 2: Display Owner Information 52**

**4.3.6 Trigger 1: Check Owner Age 53**

**4.3.7 Trigger 2: Check Appointment Date Time 54**

**4.3.8 Report 1: Summary Report of Treatment Revenue in Selected Year 56**

**4.3.9 Report 2: Detail Report of Medic Performance 63**

**4.3.10 Report 3: On-demand Report of the Pet Treatment Detail 70**

**4.4 (Nigel Lee Jian Hsee) 75**

**4.2.1 Query 1: Top Three Pets Customer’s Postcode for each Branch based on total**

**transaction amount (Strategic) 75**

**4.4.2 Query 2: Rank top treatment, to medicine sales and total revenue of a**

**branch 77**

**4.4.3 Query 3: View Customer Transaction History by entering Customer ID 80**

**4.4.4 Procedure 1: Create Transaction 84**

**4.4.5 Procedure 2: Add Transaction Detail 89**

**4.4.6 Procedure 3: Edit Transaction Detail 94**

**4.4.7 Trigger 1: Validate Payment Date Time 100**

**4.4.8 Trigger 2: Update medicine quantity and transaction total amount after**

**editing transaction detail 102**

**4.4.9 Trigger 3: Update the stock and transaction total amount after deleting**

**transaction detail 103**

**4.4.10 Trigger 4: Update total amount after inserting new transaction detail 103**

**4.4.11 Report 1: Summary report of Veterinarian Performance in a month 104**

**4.4.12 Report 2: On demand report of days that performed poorly for a specific**

**with a date range 110**

**4.4.13 Report 3: Detail report of each Branch’s Performance in a specific year 121**

**Chapter 5 Extra Effort Highlights 131**

**5.1 (Tan Yi Hong) 131**

**5.1.1 Views 131**

**5.1.2 User Defined Exception 133**

**5.1.3 Sequences 134**

**5.1.4 Triggers 134**

**5.2 (Tan Teoh Xin Ee) 135**

**5.2.1 Views 135**

**5.2.2 User Defined Exception 136**

**5.2.3 Sequence 136**

**5.2.4 Trigger 137**

**5.3 (Tan Wei Siong) 138**

**5.3.1 User Defined Exceptions 138**

**5.3.2 Sequence 138**

**5.3.3 Procedure 138**

**5.3.4 Trigger 139**

**5.4 (Nigel Lee Jian Hsee) 140**

**5.4.1 Views 140**

**5.4.2 User Defined Exceptions 141**

**5.4.3 Trigger 144**

**Chapter 1 Background of the System**

The 4 Golden Duck Wellness Veterinary Clinic is a pet clinic operating from the year 2019 with a database of 4 main functions. 4 Golden Duck Wellness Veterinary Clinic has 3 three branches which are located in Kuala Lumpur, Penang, and Kedah.

**Payment & Transaction**

This function is to record every detail of a pet owner’s transaction in the Wellness Veterinary Clinic. It can be used to track and monitor transactions of each pet owner. By recording all transactions, various reports can be generated to gain business insights. This function is able to calculate the total amount of a transaction. The transaction will include all the quantities of a medicine bought by the pet owner and treatment for their pets.

**Appointment/Booking**

The purpose of this function is to let customers make appointments for the treatment of their pets. This function can record all appointments made to make sure the service of treatment does not clash with each other and arrange the time of appointment schedule. Staff will be handling appointments and able to see the record in the tables to serve the customers accordingly.

**Pet Registration**

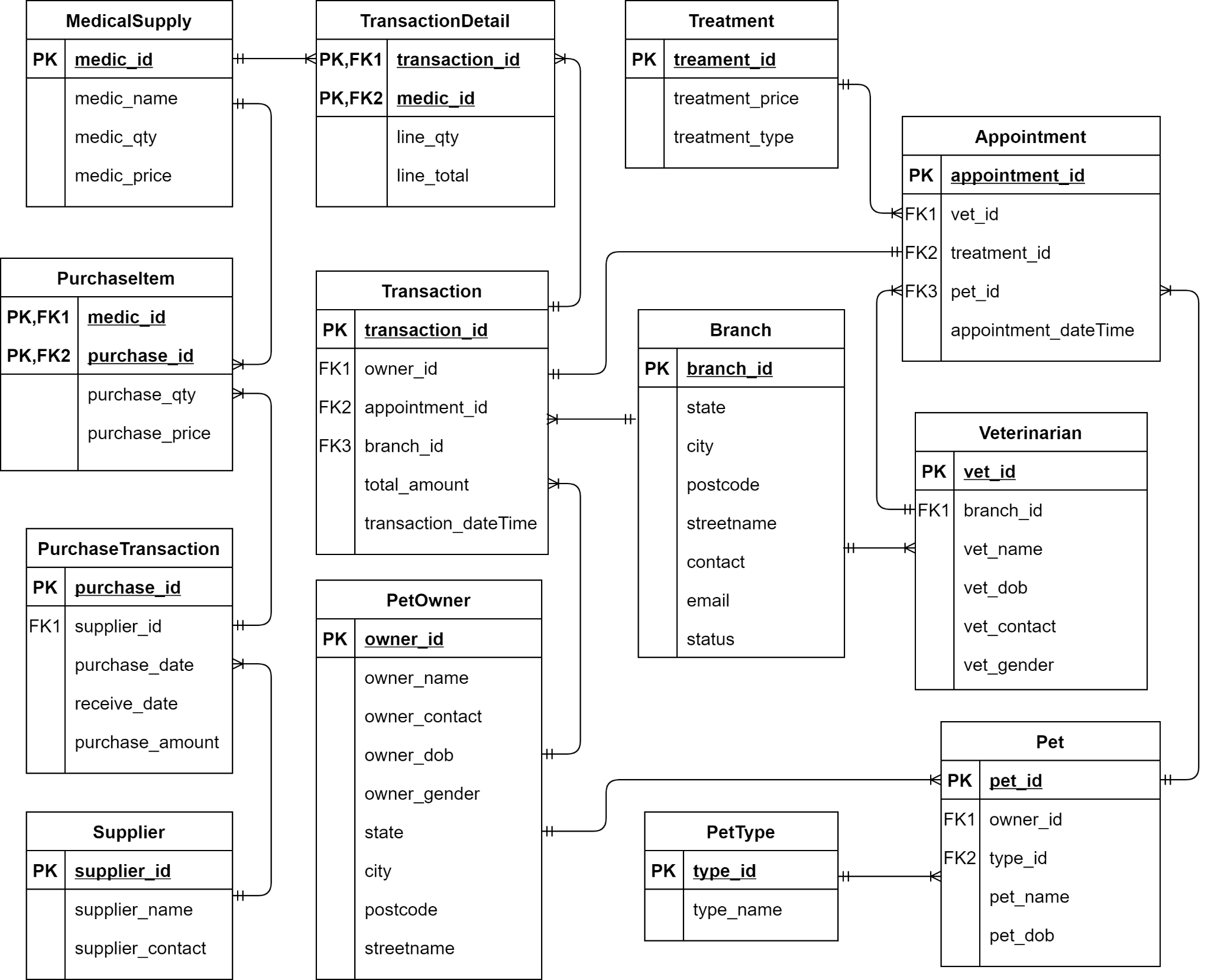
The purpose of this function is to have an easy way to register and record the pet information into the system. By recording the pet detail, it allows the staff to check whether the owner has registered before or not. If not yet registered, the system will require the staff to register the pet owner first before adding the pet.

**Medical Stock Management**

This function is mainly to manage the stock in the Wellness Veterinary Clinic such as the medicine. It can help us to keep track of the transaction of medicine from the supplier and and the transaction of medicine from the customer. This function is able to calculate the total amount of transactions for the medicine that we use to order from the supplier.

**Chapter 2 Entity-Relationship Modeling**

**2.1 ERD**

****

**2.2 Assumptions and Business Rules**

**Branch**

1. All branches of pet clinics have the same operation time which is 10am-6pm.
2. One branch can have many veterinarians, but each veterinarian can only work in one branch.(One-to-many)
3. One branch can have manytransactions, but each transaction can only have one branch.(One-to-many)

**Treatment**

1. One treatment can have many appointments, but one appointment can only have one treatment. (One-to-Many)

**Pet**

1. One pet can have many appointments, but each appointment can only have one pet.(One-to-Many)
2. Each pet can only be recorded under one pet owner, while one pet owner can have multiple pets.(Many-to-One)

**PetOwner**

1. Each pet owner can have one or many pets, while one pet can only have one pet owner. (One-to-Many)
2. Each pet owner can have one or more transactions, while each transaction can only have one pet owner recorded. (One-to-Many)

**Veterinarian**

1. One veterinarian can handle many appointments, but each appointment can only be handled by one veterinarian.(One-to-many)
2. One veterinarian can only in one branch, but one branch can have many veterinarians.(Many-to-One)

**PurchaseTransaction**

1. One PurchaseTransaction can have many PurchaseItem, but one PurchaseItem can only have one PurchaseTransaction. (One-to-Many)
2. One PurchaseTransaction can only have one Supplier, but one Supplier can have many PurchaseTransaction.

**PurchaseItem**

1. One PurchaseItem can only have one PurchaseTransaction, but one PurchaseTransaction can have many PurchaseItem. (One-to-One)

Formula

transacPurAmt = price \* quantity

**Supplier**

1. One supplier can have many PurchaseTransaction, but one PurchaseTransaction can only have one supplier. (One-to-Many)

**Medical Supply**

1. One medical supply can be included in many transaction details, but a transaction detail can have only one or none medical supply. (Zero/One-to-Many)
2. Price of the medical supply must be more than its purchase price.

**Transaction Detail**

1. One transaction detail can have one medical supply, a medical supply can be included in many transaction details.(One-to-Many)
2. One transaction detail can only be included in a transaction , a transaction can be included in many transaction details.(One-to-Many)
3. One transaction must include either one medical supply or treatment.
4. Transaction amount cannot be zero.
5. Transaction details cannot be edited after 7 days from the transaction data.

Formula

line\_total = medic\_price \* line\_qty

**Transaction**

1. Each single transaction can have many transaction details, but one transaction detail can only be included in one transaction. (One-to-Many)
2. One transaction can have only one pet owner, but a pet owner can have many transactions. (One-to-Many)
3. Each transaction can only have one appointment. (One-to-One)

Formula

total\_line\_total += line\_total

total\_amount = total\_line\_total + treatment\_price (Sum up all the line total and treatment price )

**Appointment**

1. One appointment only can have one pet, but one pet can have many appointments. (One-to-Many)
2. Appointment can only be made in business hours(10.00am to 5.00pm).
3. One appointment only can have one veterinarian, but one veterinarian can have many appointments. (One-to-Many).
4. One appointment only can have one treatment, but one treatment can be included in many appointments. (One-to-Many).
5. Each appointment can only be included in the transaction.
6. The appointment can only be made if the selected veterinarian is available on the selected period.

**Chapter 3 Data Definition**

Create table statements with appropriate constraints:

**3.1 Branch table**

CREATE TABLE Branch(

branch\_id CHAR(5) NOT NULL,

state VARCHAR2(30) NOT NULL,

city VARCHAR2(30) NOT NULL,

postcode NUMBER(5) NOT NULL,

streetName VARCHAR2(50) NOT NULL,

contact VARCHAR2(11) NOT NULL,

email VARCHAR2(30) NOT NULL,

status VARCHAR2(10) NOT NULL,

PRIMARY KEY (branch\_id),

CONSTRAINT chk\_status CHECK(status IN ('Active','Not Active'))

);

**3.2 Veterinarian table**

CREATE TABLE Veterinarian(

vet\_id CHAR(5) NOT NULL,

branch\_id CHAR(5) NOT NULL,

vet\_name VARCHAR2(30) NOT NULL,

vet\_dob DATE NOT NULL,

vet\_contact VARCHAR2(11) NOT NULL,

vet\_gender CHAR(1) NOT NULL,

PRIMARY KEY (vet\_id),

FOREIGN KEY (branch\_id) REFERENCES Branch (branch\_id),

CONSTRAINT chk\_vet\_gender CHECK(vet\_gender IN ('M','F')),

CONSTRAINT chk\_vet\_contact CHECK(REGEXP\_LIKE(vet\_contact,'^[0-9]+$'))

);

**3.3 Pet Owner table**

CREATE TABLE PetOwner(

owner\_id CHAR(5) NOT NULL,

owner\_name VARCHAR2(30) NOT NULL,

owner\_contact VARCHAR2(11) NOT NULL,

owner\_dob DATE NOT NULL,

owner\_gender CHAR(1) NOT NULL,

state VARCHAR2(30) NOT NULL,

city VARCHAR2(30) NOT NULL,

postcode VARCHAR2(5) NOT NULL,

streetName VARCHAR2(50) NOT NULL,

PRIMARY KEY (owner\_id),

CONSTRAINT chk\_owner\_gender CHECK(owner\_gender IN ('M','F')),

CONSTRAINT chk\_owner\_name CHECK(REGEXP\_LIKE(owner\_name,'^[a-z A-z]+$')),

CONSTRAINT chk\_owner\_contact CHECK(REGEXP\_LIKE(owner\_contact,'^[0-9]+$'))

);

**3.4 Pet Type table**

CREATE TABLE PetType(

type\_id CHAR(5) NOT NULL,

type\_name VARCHAR2(30) NOT NULL,

PRIMARY KEY (type\_id),

CONSTRAINT chk\_type\_name CHECK(REGEXP\_LIKE(type\_name,'^[a-z A-z]+$'))

);

**3.5 Pet table**

CREATE TABLE Pet(

pet\_id CHAR(5) NOT NULL,

owner\_id CHAR(5) NOT NULL,

pet\_name VARCHAR2(30) NOT NULL,

pet\_dob DATE NOT NULL,

type\_id CHAR(5) NOT NULL,

PRIMARY KEY (pet\_id),

FOREIGN KEY (owner\_id) REFERENCES PetOwner (owner\_id),

FOREIGN KEY (type\_id) REFERENCES PetType (type\_id),

CONSTRAINT chk\_pet\_name CHECK(REGEXP\_LIKE(pet\_name,'^[a-z A-z]+$'))

);

**3.6 Treatment table**

CREATE TABLE Treatment(

treatment\_id CHAR(5) NOT NULL,

treatment\_price NUMBER(7,2) NOT NULL,

treatment\_type VARCHAR2(50) NOT NULL,

PRIMARY KEY (treatment\_id),

CONSTRAINT chk\_treatment\_price CHECK(treatment\_price > 0)

);

**3.7 Appointment table**

CREATE TABLE Appointment(

appointment\_id CHAR(10) NOT NULL,

vet\_id CHAR(5) NOT NULL,

treatment\_id CHAR(5) NOT NULL,

pet\_id CHAR(5) NOT NULL,

appointment\_dateTime DATE NOT NULL,

PRIMARY KEY (appointment\_id),

FOREIGN KEY (vet\_id) REFERENCES Veterinarian (vet\_id),

FOREIGN KEY (treatment\_id) REFERENCES Treatment (treatment\_id),

FOREIGN KEY (pet\_id) REFERENCES Pet (pet\_id)

);

**3.8 Transaction table**

CREATE TABLE Transaction(

transaction\_id CHAR(11) NOT NULL,

owner\_id CHAR(5) NOT NULL,

appointment\_id CHAR(10) NOT NULL,

branch\_id CHAR(5) NOT NULL,

total\_amount NUMBER(7,2) NOT NULL,

transaction\_dateTime DATE NOT NULL,

PRIMARY KEY (transaction\_id),

FOREIGN KEY (owner\_id) REFERENCES PetOwner (owner\_id),

FOREIGN KEY (appointment\_id) REFERENCES Appointment (appointment\_id),

FOREIGN KEY (branch\_id) REFERENCES Branch (branch\_id),

CONSTRAINT chk\_total\_amount CHECK(total\_amount > 0)

);

**3.9 Medical Supply table**

CREATE TABLE MedicalSupply(

medic\_id CHAR(5) NOT NULL,

medic\_name VARCHAR2(30) NOT NULL,

medic\_qty NUMBER(5) NOT NULL,

medic\_price NUMBER(7,2) NOT NULL,

PRIMARY KEY(medic\_id),

CONSTRAINT chk\_qty CHECK(medic\_qty>=0),

CONSTRAINT chk\_price CHECK(medic\_price>0)

);

**3.10 Transaction Detail table**

CREATE TABLE TransactionDetail(

transaction\_id CHAR(11) NOT NULL,

medic\_id CHAR(5) NOT NULL,

line\_qty NUMBER(3) NOT NULL,

line\_total Number(7,2) NOT NULL,

PRIMARY KEY (transaction\_id, medic\_id),

FOREIGN KEY (transaction\_id) REFERENCES Transaction (transaction\_id),

FOREIGN KEY (medic\_id) REFERENCES MedicalSupply (medic\_id),

CONSTRAINT chk\_line\_total CHECK(line\_total > 0),

CONSTRAINT chk\_line\_qty CHECK(line\_qty > 0)

);

**3.11 Supplier table**

CREATE TABLE Supplier(

supplier\_id CHAR(5) NOT NULL,

supplier\_name VARCHAR2(50) NOT NULL,

supplier\_contact VARCHAR2(11) NOT NULL,

PRIMARY KEY (supplier\_id),

CONSTRAINT chk\_sup\_name CHECK(REGEXP\_LIKE(supplier\_name,'^[a-z A-z]+$')),

CONSTRAINT chk\_sup\_contact CHECK(REGEXP\_LIKE(supplier\_contact,'^[0-9]+$'))

);

**3.12 Purchase Transaction table**

CREATE TABLE PurchaseTransaction(

purchase\_id CHAR(5) NOT NULL,

supplier\_id CHAR(5) NOT NULL,

purchase\_date DATE NOT NULL,

receive\_date DATE NOT NULL,

purchase\_amount NUMBER(7,2) NOT NULL,

PRIMARY KEY (purchase\_id),

FOREIGN KEY (supplier\_id) REFERENCES Supplier (supplier\_id),

CONSTRAINT chk\_pur\_amount CHECK(purchase\_amount>= 0)

);

**3.13 Purchase Item table**

CREATE TABLE PurchaseItem(

medic\_id CHAR(5) NOT NULL,

purchase\_id CHAR(5) NOT NULL,

purchase\_qty NUMBER(3) NOT NULL,

purchase\_price NUMBER(7,2) NOT NULL,

PRIMARY KEY (medic\_id, purchase\_id),

FOREIGN KEY (medic\_id) REFERENCES MedicalSupply (medic\_id),

FOREIGN KEY (purchase\_id) REFERENCES PurchaseTransaction (purchase\_id),

CONSTRAINT chk\_pqty CHECK(purchase\_qty>0),

CONSTRAINT chk\_pprice CHECK(purchase\_price>0)

);

**Chapter 4 Queries, Procedures, Triggers and Reports**

**4.1 (Tan Yi Hong)**

**4.1.1 Query 1: Top pet type that received treatment in each branch (Strategic)**

**Purpose: The purpose of this query is to let the clinic know the top pet type that received treatment in each branch so that the organization can focus more on the service on which type of pets in each branch to provide better service to their customer for all branches.**

clear break

clear compute

set linesize 80

set pagesize 100

break on state on branch\_id skip 1

COMPUTE SUM LABEL TOTAL OF nooftreatment percentage transactionamount on branch\_id

TTITLE ON

TTITLE CENTER 'Top Pet Type that received treatment in each branch' SKIP 1-

CENTER ========================================================= SKIP 2

COLUMN branch\_id FORMAT a10

COLUMN branch\_id HEADING 'Branch ID'

COLUMN state FORMAT a15

COLUMN type\_name FORMAT a10

COLUMN type\_name HEADING 'Pet Type'

COLUMN nooftreatment HEADING 'Treatment|Received'

COLUMN transactionamount FORMAT 9999999.99

COLUMN transactionamount HEADING 'Total|Transaction|Made'

COLUMN Percentage FORMAT 999.99

COLUMN Percentage HEADING 'Percent|Over|Total'

CREATE OR REPLACE VIEW topPetTreatment AS

SELECT t.branch\_id, pt.type\_name, COUNT(t.appointment\_id) AS NoOfTreatment, SUM(t.total\_amount) AS TransactionAmount

FROM appointment a, veterinarian v, pet p, petType pt, transaction t

WHERE t.appointment\_id=a.appointment\_id AND a.vet\_id=v.vet\_id AND a.pet\_id=p.pet\_id

AND p.type\_id=pt.type\_id AND t.appointment\_id=a.appointment\_id

GROUP BY t.branch\_id, pt.type\_name

ORDER BY t.branch\_id, SUM(t.total\_amount) DESC;

SELECT a.branch\_id, b.state, a.type\_name, a.nooftreatment, (a.NoOfTreatment/COUNT(t.appointment\_id))\*100 AS Percentage, a.transactionAmount, RANK() OVER(PARTITION BY a.branch\_id ORDER BY a.transactionAmount DESC) Ranks

FROM topPetTreatment a, transaction t, branch b

WHERE a.branch\_id=t.branch\_id AND a.branch\_id=b.branch\_id

GROUP BY a.branch\_id, b.state, a.type\_name, a.nooftreatment, a.transactionAmount

ORDER BY a.branch\_id, a.transactionamount DESC;

**Sample Output:**

****

**4.1.2 Query 2: Appointment made on times of a day in each branch for last year (Tactical)**

**Purpose: The purpose of this query is to list out all appointments made during the times of a day in each branch and to know the peak business time to let the organization adjust the shifts between the veterinarian to suitable shifts whether increasing or decreasing the time shifts between them.**

clear break

clear compute

set linesize 71

set pagesize 100

BREAK ON REPORT

COMPUTE SUM LABEL TOTAL AVG LABEL AVERAGE OF MORNING AFTERNOON EVENING totalappointment ON REPORT

COLUMN morning FORMAT 999999999

COLUMN afternoon FORMAT 999999999

COLUMN evening FORMAT 999999999

TTITLE ON

TTITLE CENTER 'Appointment made on times of a day in each branch for last year' SKIP 1-

CENTER =================================================================== SKIP 2

COLUMN branch\_id FORMAT a10

COLUMN branch\_id HEADING 'Branch ID'

COLUMN state FORMAT a15

COLUMN totalappointment FORMAT 999999999

COLUMN totalappointment HEADING 'Total|Appointment'

CREATE OR REPLACE VIEW morningApp AS

SELECT t.branch\_id, COUNT(t.transaction\_id) AS MORNING

FROM appointment a, transaction t

WHERE t.appointment\_id = a.appointment\_id AND

EXTRACT(HOUR FROM CAST(a.appointment\_datetime AS TIMESTAMP)) BETWEEN 10 AND 12

AND EXTRACT(YEAR FROM a.appointment\_datetime) = EXTRACT(YEAR FROM SYSDATE)-1

GROUP BY t.branch\_id

ORDER BY t.branch\_id;

CREATE OR REPLACE VIEW afternoonApp AS

SELECT t.branch\_id, COUNT(t.transaction\_id) AS AFTERNOON

FROM appointment a, transaction t

WHERE t.appointment\_id = a.appointment\_id AND

EXTRACT(HOUR FROM CAST(a.appointment\_datetime AS TIMESTAMP)) BETWEEN 13 AND 15

AND EXTRACT(YEAR FROM a.appointment\_datetime) = EXTRACT(YEAR FROM SYSDATE)-1

GROUP BY t.branch\_id

ORDER BY t.branch\_id;

CREATE OR REPLACE VIEW eveningApp AS

SELECT t.branch\_id, COUNT(t.transaction\_id) AS EVENING

FROM appointment a, transaction t

WHERE t.appointment\_id = a.appointment\_id AND

EXTRACT(HOUR FROM CAST(a.appointment\_datetime AS TIMESTAMP)) BETWEEN 16 AND 18

AND EXTRACT(YEAR FROM a.appointment\_datetime) = EXTRACT(YEAR FROM SYSDATE)-1

GROUP BY t.branch\_id

ORDER BY t.branch\_id;

SELECT a.branch\_id, d.state, a.morning, b.afternoon, c.evening, (a.morning + b.afternoon + c.evening) AS TotalAppointment

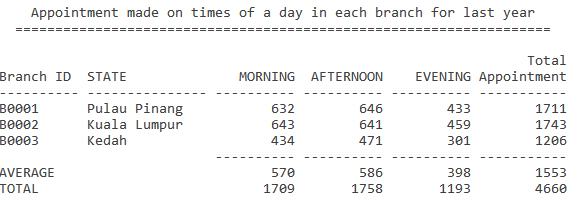
FROM morningApp a, afternoonApp b, eveningApp c, branch d

WHERE a.branch\_id=b.branch\_id AND a.branch\_id=c.branch\_id AND a.branch\_id=d.branch\_id

GROUP BY a.branch\_id, d.state, a.morning, b.afternoon, c.evening

ORDER BY a.branch\_id;

**Sample Output:**

****

**4.1.3 Query 3: Year 2020 first half sales vs second half sales in each branch (Operational)**

**Purpose: The purpose of this query is to calculate and compare the sales of first half and second half in the year 2020 for each branch. It can show the operational sales of the clinic and compare the sales difference or percent difference to better understand the trends of their sales growth.**

clear break

clear compute

BREAK ON REPORT

COMPUTE SUM LABEL TOTAL AVG LABEL AVERAGE OF SALES2020\_1STHALF SALES2020\_2NDHALF SALESDIFF ON REPORT

set linesize 95

set pagesize 100

TTITLE ON

TTITLE CENTER 'Year 2020 first half sales vs second half sales' SKIP 1-

CENTER ================================================== SKIP 2

COLUMN branch\_id FORMAT a10

COLUMN branch\_id HEADING 'Branch ID'

COLUMN state FORMAT a15

COLUMN SALES2020\_1STHALF FORMAT 9999999.99

COLUMN SALES2020\_1STHALF HEADING 'First Half Sales'

COLUMN SALES2020\_2NDHALF FORMAT 9999999.99

COLUMN SALES2020\_2NDHALF HEADING 'Second Half Sales'

COLUMN SALESDIFF FORMAT 9999999.99

COLUMN SALESDIFF HEADING 'Sales Different'

COLUMN SALESDIFF\_PERCENTAGE FORMAT 999.99

COLUMN SALESDIFF\_PERCENTAGE HEADING 'Percent Different'

CREATE OR REPLACE VIEW Sales2020\_1stHalf AS

SELECT branch\_id, SUM(total\_amount) AS Sales2020\_1stHalf

FROM transaction

WHERE EXTRACT(YEAR FROM transaction\_dateTime) = 2020 AND EXTRACT(MONTH FROM transaction\_dateTime) <= 6

GROUP BY branch\_id

ORDER BY branch\_id, SUM(total\_amount) DESC;

CREATE OR REPLACE VIEW Sales2020\_2ndHalf AS

SELECT branch\_id, SUM(total\_amount) AS Sales2020\_2ndHalf

FROM transaction

WHERE EXTRACT(YEAR FROM transaction\_dateTime) = 2020 AND EXTRACT(MONTH FROM transaction\_dateTime) > 6

GROUP BY branch\_id

ORDER BY branch\_id, SUM(total\_amount) DESC;

SELECT a.branch\_id, c.state, a.Sales2020\_1stHalf, b.Sales2020\_2ndHalf, Sales2020\_2ndHalf-Sales2020\_1stHalf AS SalesDiff, (Sales2020\_2ndHalf/Sales2020\_1stHalf)\*100 AS SalesDiff\_Percentage

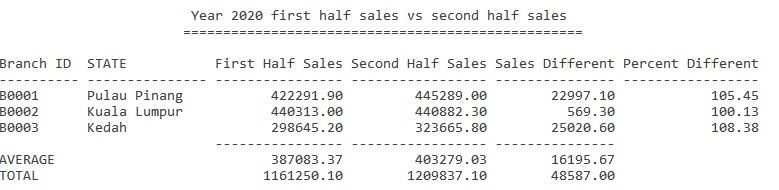
FROM branch c, Sales2020\_1stHalf a, Sales2020\_2ndHalf b

WHERE a.branch\_id=c.branch\_id AND b.branch\_id=c.branch\_id

GROUP BY a.branch\_id, c.state, a.Sales2020\_1stHalf, b.Sales2020\_2ndHalf

ORDER BY branch\_id;

**Sample Output:**



**4.1.4 Procedure 1: Add Appointment record**

**Purpose: The purpose of this stored procedure is to add a new appointment record into the database when customers want to make new appointments.**

CREATE OR REPLACE PROCEDURE PRC\_ADD\_APPOINTMENT(IN\_vetID in CHAR, IN\_treatmentID in CHAR, IN\_petID in CHAR, IN\_dateTime in DATE) AS

v\_insertID CHAR(10);

v\_branchID CHAR(5);

counter\_t NUMBER;

counter\_p NUMBER;

e\_invalid\_treatment EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_treatment, -20050);

e\_invalid\_pet EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_pet, -20051);

BEGIN

counter\_t := 0;

counter\_p := 0;

SELECT branch\_id INTO v\_branchID

FROM veterinarian

WHERE vet\_id = IN\_vetID;

SELECT COUNT(\*) INTO counter\_t

FROM treatment

WHERE treatment\_id = IN\_treatmentID;

IF counter\_t = 0 THEN

RAISE\_APPLICATION\_ERROR(-20050, 'Invalid Treatment ID.');

END IF;

SELECT COUNT(\*) INTO counter\_p

FROM pet

WHERE pet\_id = IN\_petID;

IF counter\_p = 0 THEN

RAISE\_APPLICATION\_ERROR(-20051, 'Invalid Pet ID.');

END IF;

IF v\_branchID = 'B0001' THEN

v\_insertID := TO\_CHAR('PP'||app\_seq\_PG.NEXTVAL);

ELSIF v\_branchID = 'B0002' THEN

v\_insertID := TO\_CHAR('KL'||app\_seq\_KL.NEXTVAL);

ELSIF v\_branchID = 'B0003' THEN

v\_insertID := TO\_CHAR('KD'||app\_seq\_KD.NEXTVAL);

END IF;

insert into appointment values(v\_insertID,IN\_vetID,IN\_treatmentID,IN\_petID,IN\_dateTime);

DBMS\_OUTPUT.PUT\_LINE (CHR(10));

DBMS\_OUTPUT.PUT\_LINE ('Appointment add SUCCESSFUL as follows: ');

DBMS\_OUTPUT.PUT\_LINE ('Appointment ID: '||v\_insertID||'|Veterinarian ID: '||IN\_vetID||'|Treatment ID: '||IN\_treatmentID||'|Pet ID: '||IN\_petID||'|Appointment Date Time: '||IN\_dateTime);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE ('No Veterinarian found');

WHEN e\_invalid\_treatment THEN

DBMS\_OUTPUT.PUT\_LINE('No such Treatment ID');

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

WHEN e\_invalid\_pet THEN

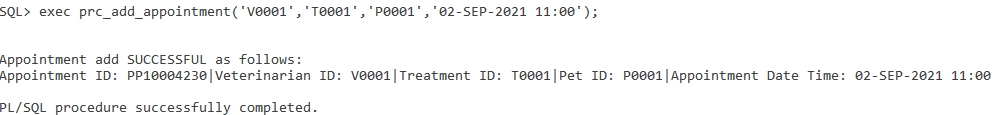
DBMS\_OUTPUT.PUT\_LINE('No such Pet ID');

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

**Sample Output:**

****

**4.1.5 Procedure 2: Delete Appointment record**

**Purpose: The purpose of this stored procedure is to delete an existing record from the appointment table when a customer has cancelled the appointment made. It will only require the appointment ID to delete the record.**

CREATE OR REPLACE PROCEDURE PRC\_DEL\_APPOINTMENT(IN\_appointmentID in CHAR) AS

BEGIN

DELETE FROM appointment

WHERE appointment\_id = IN\_appointmentID;

DBMS\_OUTPUT.PUT\_LINE (IN\_appointmentID||' Deleted successfully.');

EXCEPTION

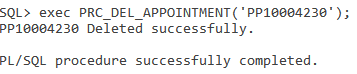
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE ('No Appointment found');

END;

/

**Sample Output:**

****

**4.1.6 Trigger 1: Validate the proper date time on insertion of appointment**

**Purpose: The purpose of this trigger is to validate the newly inserted appointment to match in a proper date and time which can make an appointment.**

CREATE OR REPLACE TRIGGER trg\_appointmentDateTime

BEFORE INSERT OR UPDATE ON Appointment

FOR EACH ROW

BEGIN

IF :new.appointment\_datetime<SYSDATE THEN

RAISE\_APPLICATION\_ERROR(-20052, 'Cannot insert the date time before now.' );

ELSIF EXTRACT(HOUR FROM CAST(:new.appointment\_datetime AS TIMESTAMP)) < 10 THEN

RAISE\_APPLICATION\_ERROR(-20053, 'Date time must be after business hour.' );

ELSIF EXTRACT(HOUR FROM CAST(:new.appointment\_datetime AS TIMESTAMP)) > 17 THEN

RAISE\_APPLICATION\_ERROR(-20054, 'Date time must be before business hour.' );

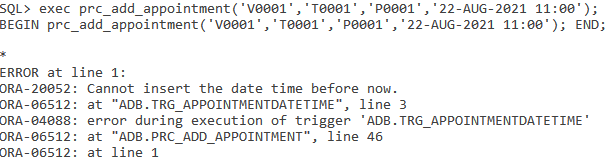
END IF;

END;

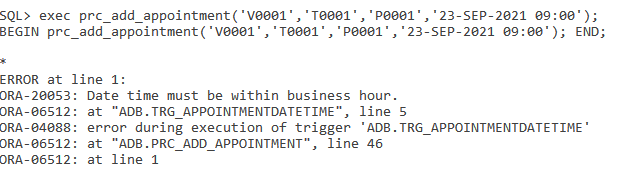
/

**Sample Output:**

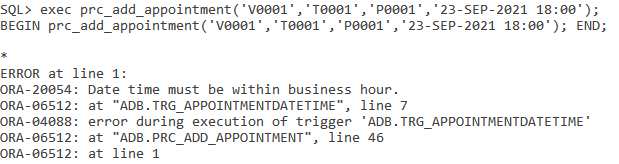
**When the appointment insert is before now:**

****

**When the appointment insert is before the business hour:**

****

**When the appointment insert is after the business hour:**

****

**4.1.7 Trigger 2: Monitor the deletion of appointment record**

**Purpose: The purpose of this trigger is to check whether the appointment that user wants to delete is recorded in the transaction or not, and it will be unable to delete when the record is in the transaction already.**

CREATE OR REPLACE TRIGGER trg\_delAppointment

BEFORE DELETE ON Appointment

FOR EACH ROW

DECLARE

counter NUMBER;

BEGIN

counter := 0;

SELECT COUNT(\*) INTO counter

FROM transaction

WHERE appointment\_id = :old.appointment\_id;

IF counter = 1 THEN

DBMS\_OUTPUT.PUT\_LINE(:old.appointment\_id||' has been recorded in the Transaction and cannot be deleted');

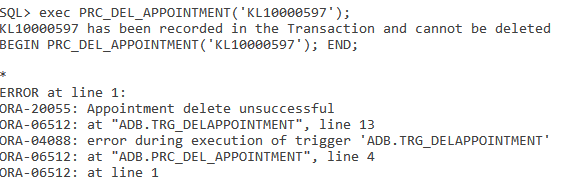
RAISE\_APPLICATION\_ERROR(-20055,'Appointment delete unsuccessful');

END IF;

END;

/

**Sample Output:**

****

**4.1.8 Report 1: Summary report of Specific veterinarian with all of his/her transactions done in a year**

**Purpose: The purpose of this report is to summarize a specific veterinarian with all of his/her transactions done in a selected year with the form of a detailed report. It can let organizations know the contribution of the transaction amount they made or to trace the total amount of transactions done by that veterinarian.**

CREATE OR REPLACE PROCEDURE prc\_vet\_summary(IN\_vetID in CHAR, IN\_year in NUMBER) AS

v\_vetName VARCHAR2(50);

v\_branchID CHAR(5);

v\_state VARCHAR2(50);

v\_totalAmount NUMBER;

counter NUMBER;

record\_count NUMBER;

e\_norecord EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_norecord,-20060);

CURSOR vet\_trans IS

SELECT t.transaction\_id, t.appointment\_id, tr.treatment\_type, t.transaction\_dateTime, t.total\_amount

FROM transaction t, appointment a, treatment tr

WHERE t.appointment\_id=a.appointment\_id AND a.treatment\_id=tr.treatment\_id AND a.vet\_id=IN\_vetID AND EXTRACT(YEAR FROM t.transaction\_dateTime) = IN\_year

ORDER BY transaction\_dateTime DESC;

BEGIN

v\_totalAmount := 0;

counter := 0;

record\_count := 0;

SELECT COUNT(\*) INTO record\_count

FROM transaction t, appointment a

WHERE t.appointment\_id=a.appointment\_id AND EXTRACT(YEAR FROM t.transaction\_dateTime) = IN\_year AND a.vet\_id=IN\_vetID;

IF record\_count = 0 THEN

RAISE\_APPLICATION\_ERROR(-20060,'No record found');

END IF;

SELECT v.vet\_name, v.branch\_id, b.state INTO v\_vetName, v\_branchID, v\_state

FROM veterinarian v, branch b

WHERE v.branch\_id=b.branch\_id AND vet\_id=IN\_vetID;

DBMS\_OUTPUT.PUT\_LINE (chr(10));

DBMS\_OUTPUT.PUT\_LINE ('Summary report of all transaction made by Veterinarian '||IN\_vetID);

DBMS\_OUTPUT.PUT\_LINE('Report generated on : ' || TO\_CHAR(CURRENT\_DATE, 'DD-MM-YYYY HH:MI:SS') || ' by ' || USER);

DBMS\_OUTPUT.PUT\_LINE (chr(10));

DBMS\_OUTPUT.PUT\_LINE ('Veterinarian Name : '||v\_vetName);

DBMS\_OUTPUT.PUT\_LINE ('Branch ID : '||v\_branchID);

DBMS\_OUTPUT.PUT\_LINE ('State : '||v\_state);

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 120, '-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Transaction ID', 23, ' ') || RPAD('Appointment ID', 23, ' ') || RPAD('Treatment Type', 30, ' ')|| RPAD('Transaction Date Time', 32, ' ') || RPAD('Total Amount', 20, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 120, '-'));

FOR trans IN vet\_trans LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD(trans.transaction\_id,23,' ')||RPAD(trans.appointment\_id,23,' ')|| RPAD(trans.treatment\_type, 30, ' ')||RPAD(trans.transaction\_dateTime, 32, ' ') ||'RM '|| RPAD(TRIM(TO\_CHAR(trans.total\_amount,'999G999D99')), 17, ' '));

v\_totalAmount := v\_totalAmount + trans.total\_amount;

counter := counter + 1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 120, '='));

DBMS\_OUTPUT.PUT\_LINE(RPAD(('Total Treatment Done : '||counter),75,' ')||'Total Amount of Transaction : RM '||TRIM(TO\_CHAR(v\_totalAmount,'999G999D99')));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 120, '='));

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE ('No Veterinarian found');

WHEN e\_norecord THEN

DBMS\_OUTPUT.PUT\_LINE('--------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Failed to print report for ' || IN\_year || '.');

DBMS\_OUTPUT.PUT\_LINE('--------------------------------');

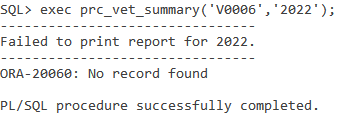
DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

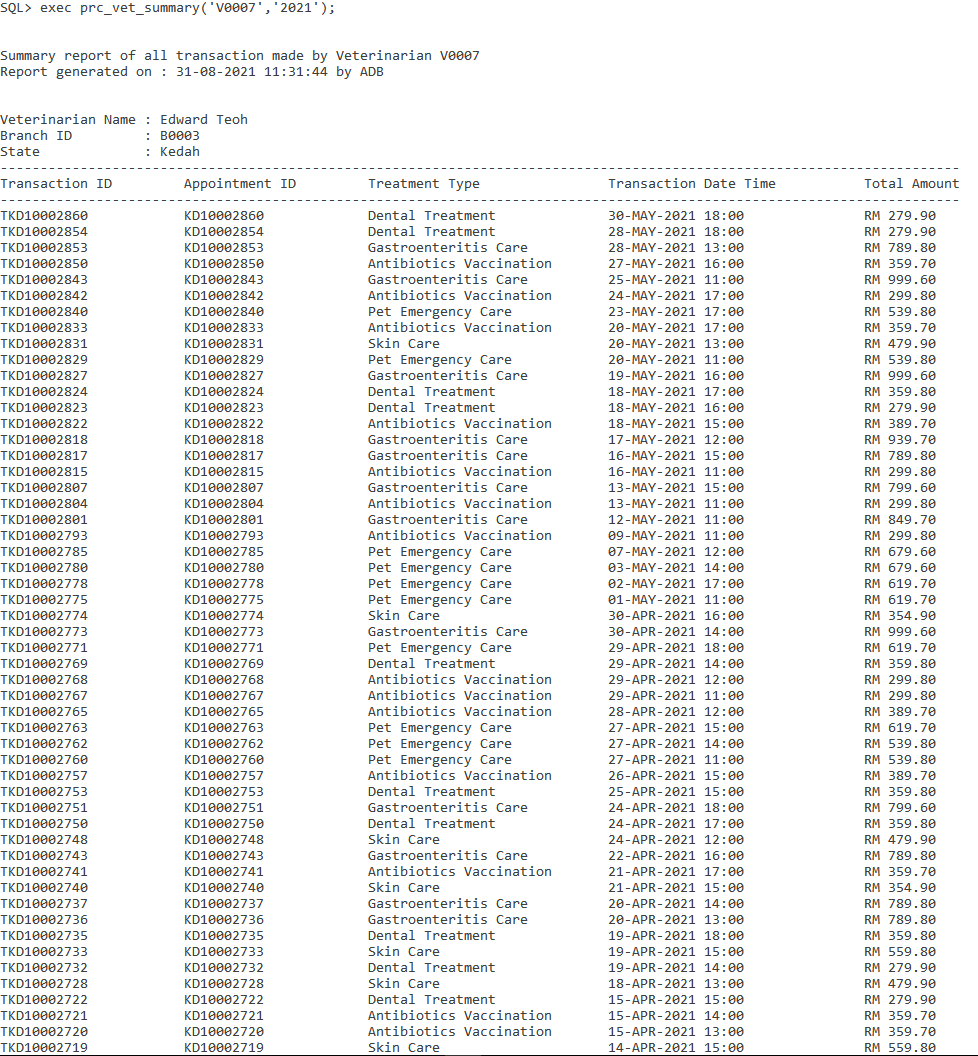
END;

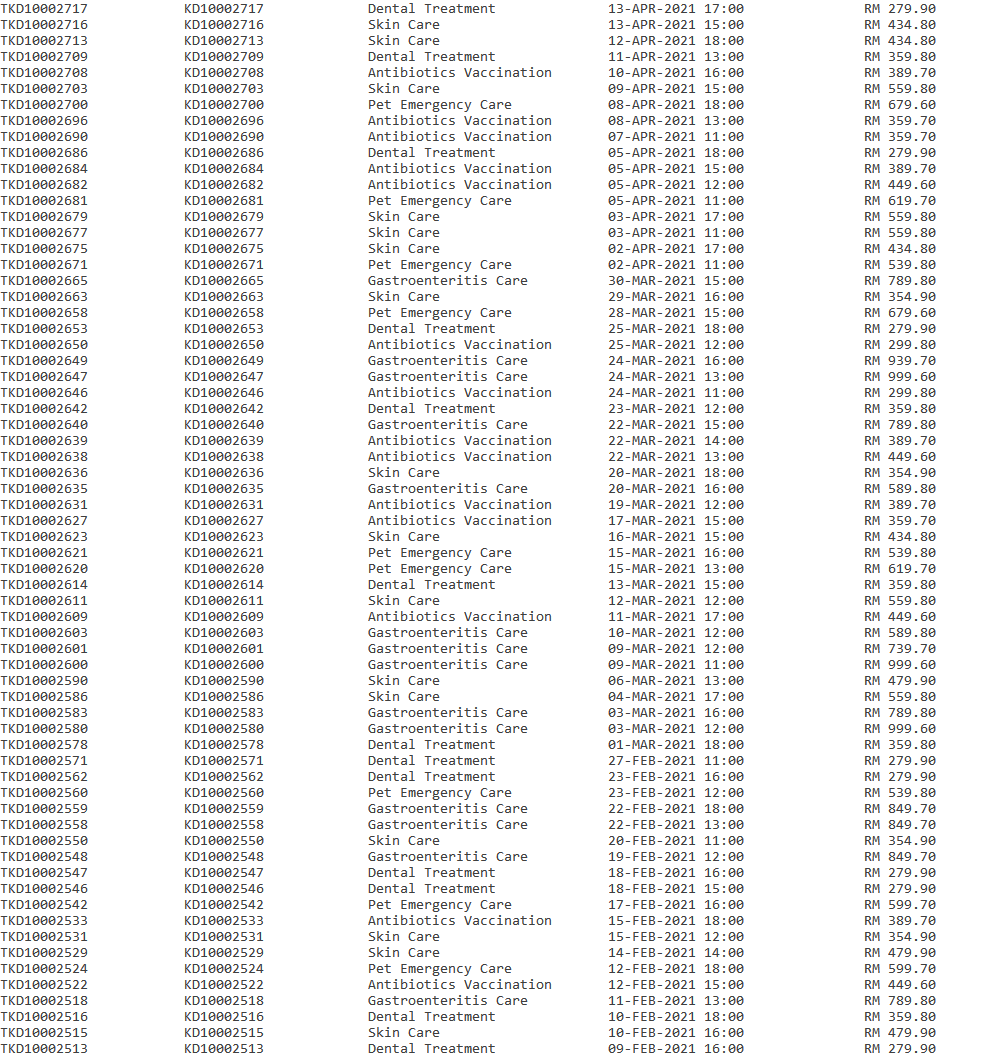
/

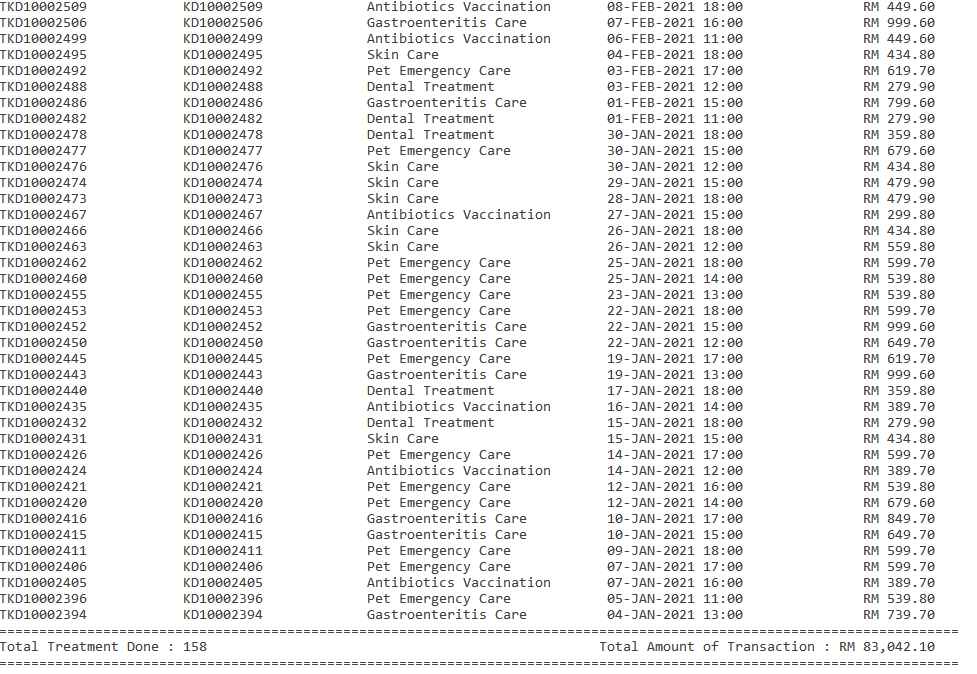
exec prc\_vet\_summary('V0007','2021');

**Sample Output:**

****

****

****

****

**4.1.9 Report 2: Detail report of Customer list in specific state with appointment made in a year**

**Purpose: The purpose of this report is to list all customers with all of their appointments made in a specific state, in a selected year along with their details. It will provide in detail every appointment made by the customers in a year. Organizations in the clinic can view all the appointments made by every customer to see their potential loyal customers.**

CREATE OR REPLACE PROCEDURE prc\_less\_appointment(IN\_state IN VARCHAR2,IN\_year IN NUMBER) AS

counter NUMBER;

record\_count NUMBER;

e\_norecord EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_norecord,-20062);

CURSOR cust\_cursor IS

SELECT owner\_id, owner\_name, owner\_gender, owner\_contact, state

FROM petowner

WHERE state = IN\_state;

CURSOR app\_cursor IS

SELECT t.owner\_id, a.appointment\_id, a.pet\_id, p.pet\_name, pt.type\_name, a.treatment\_id, tr.treatment\_type, a.vet\_id, v.vet\_name, a.appointment\_dateTime

FROM appointment a, transaction t, treatment tr, veterinarian v, pet p, pettype pt

WHERE t.appointment\_id=a.appointment\_id AND a.treatment\_id=tr.treatment\_id AND a.vet\_id=v.vet\_id AND a.pet\_id=p.pet\_id AND p.type\_id=pt.type\_id AND EXTRACT(YEAR FROM a.appointment\_dateTime) = IN\_year;

BEGIN

record\_count := 0;

SELECT COUNT(\*) INTO record\_count

FROM transaction t, appointment a, branch b

WHERE t.appointment\_id=a.appointment\_id AND EXTRACT(YEAR FROM a.appointment\_dateTime) = IN\_year AND b.state LIKE IN\_state;

IF record\_count = 0 THEN

RAISE\_APPLICATION\_ERROR(-20062,'No record found');

END IF;

DBMS\_OUTPUT.PUT\_LINE (chr(10));

DBMS\_OUTPUT.PUT\_LINE ('All customers in '||IN\_state||' with appointment made in the year '||IN\_year);

DBMS\_OUTPUT.PUT\_LINE('Report generated on : ' || TO\_CHAR(CURRENT\_DATE, 'DD-MM-YYYY HH:MI:SS') || ' by ' || USER);

DBMS\_OUTPUT.PUT\_LINE (chr(10));

FOR cust IN cust\_cursor LOOP

counter := 0;

DBMS\_OUTPUT.PUT\_LINE ('Customer ID : '||cust.owner\_ID);

DBMS\_OUTPUT.PUT\_LINE ('Customer Name : '||cust.owner\_name);

DBMS\_OUTPUT.PUT\_LINE ('Contact : '||cust.owner\_contact);

DBMS\_OUTPUT.PUT\_LINE ('Gender : '||cust.owner\_gender);

DBMS\_OUTPUT.PUT\_LINE ('State : '||cust.state);

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 135, '-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Appointment ID', 20, ' ') || RPAD('Pet', 28, ' ') || RPAD('Treatment', 35, ' ')|| RPAD('Veterinarian Handled', 31, ' ') || RPAD('Appointment Date Time', 25, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 135, '-'));

FOR app IN app\_cursor LOOP

IF app.owner\_ID = cust.owner\_ID THEN

DBMS\_OUTPUT.PUT\_LINE(RPAD(app.appointment\_id, 20, ' ') || RPAD((app.pet\_id||' '||app.pet\_name||' ( '||app.type\_name||')'), 28, ' ') || RPAD((app.treatment\_id||' '||app.treatment\_type), 35, ' ')|| RPAD((app.vet\_id||' '||app.vet\_name), 31, ' ') || RPAD(app.appointment\_dateTime, 20, ' '));

counter := counter + 1;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 135, '='));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*',113,' ')||'No of record found: '||counter);

DBMS\_OUTPUT.PUT\_LINE(CHR(10));

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE ('No Veterinarian found');

WHEN e\_norecord THEN

DBMS\_OUTPUT.PUT\_LINE('---------------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('No record found for state '||IN\_state||' in year '||IN\_year||'.');

DBMS\_OUTPUT.PUT\_LINE('---------------------------------------------------');

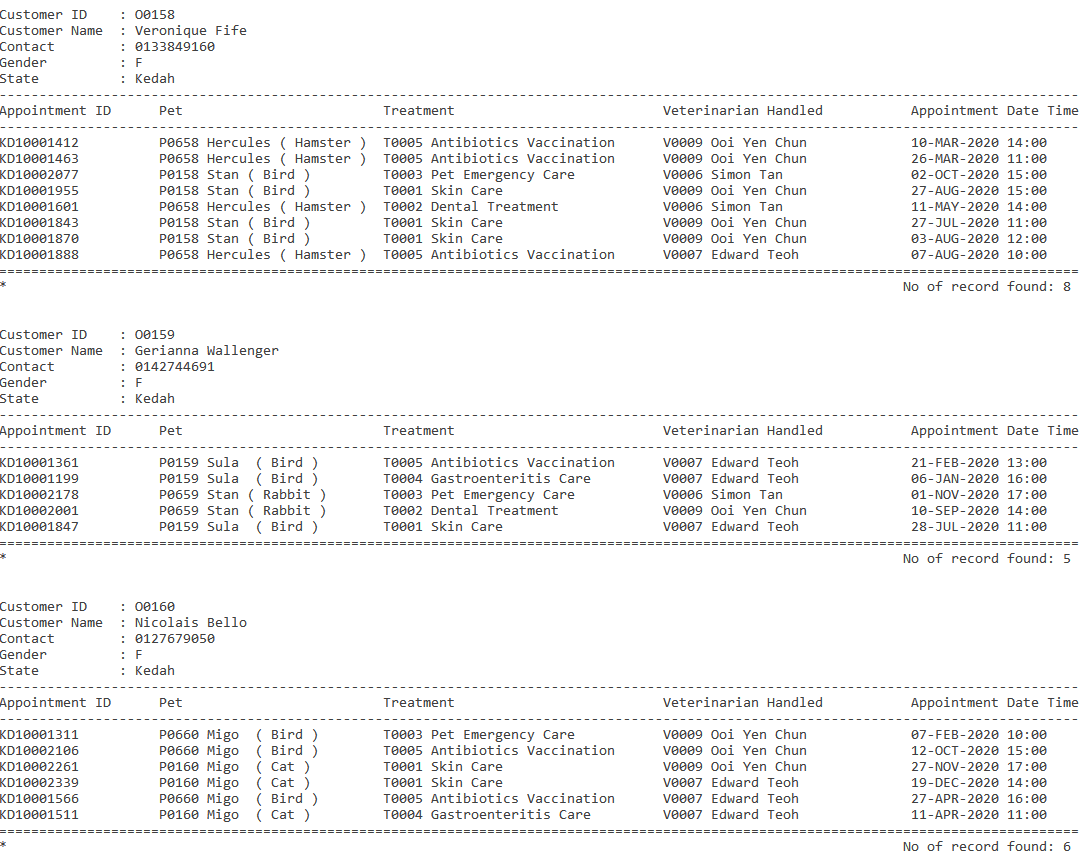
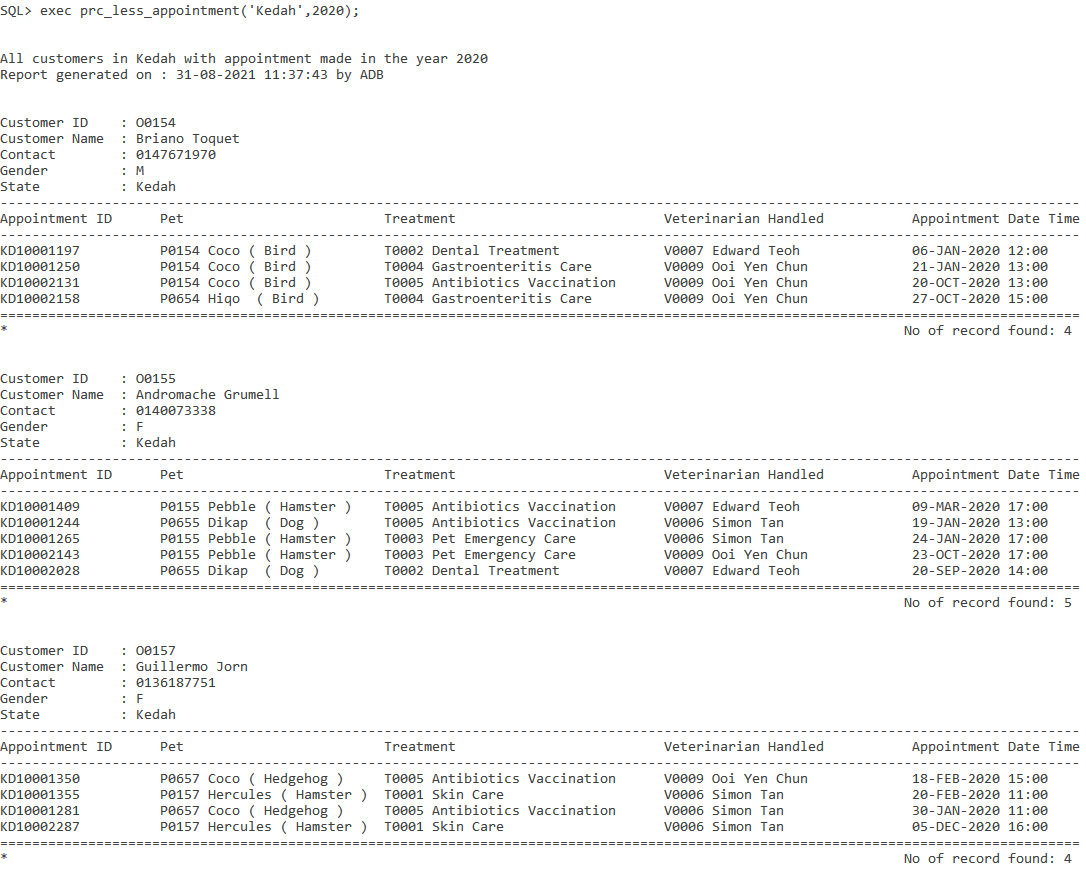
DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

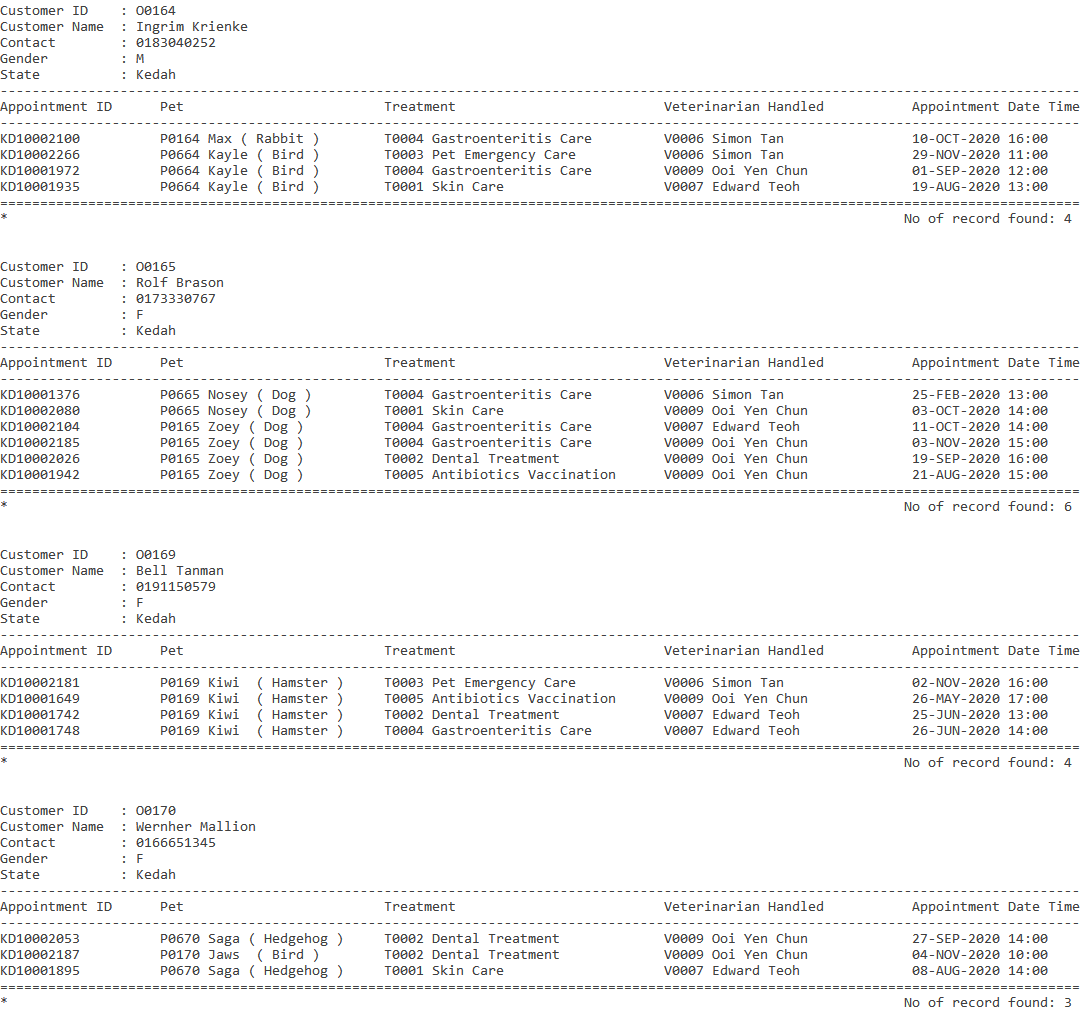
END;

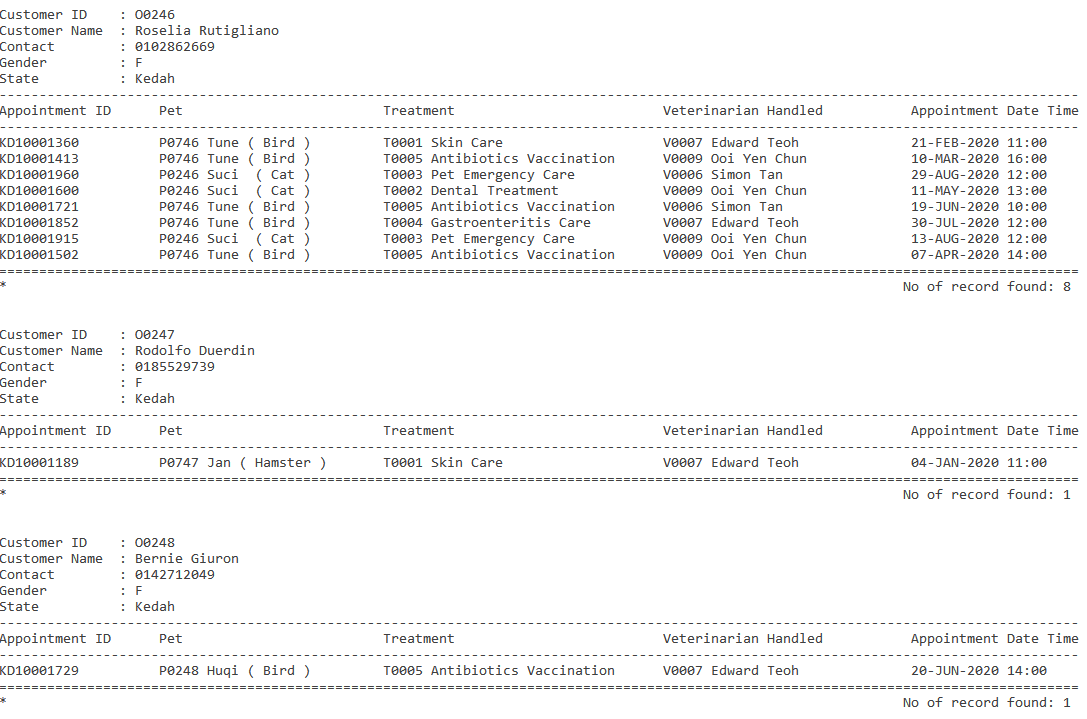
/

exec prc\_less\_appointment('Kedah',2020);

**Sample Output:**

****

****

****

**4.1.10 Report 3: On demand report of Customer list who do transaction for less than three times in a year**

**Purpose: The purpose of this report is to list out the customers who have less than 3 transactions done in a year. This can let the management to predict the churn customer by viewing this report which also includes the last transaction made by the customer. In this way, they can try to ask the listed customers for their feedback and satisfaction against the clinic to know why they will be churning.**

CREATE OR REPLACE PROCEDURE prc\_less\_appointment(IN\_year IN NUMBER) AS

counter NUMBER;

record\_count NUMBER;

e\_norecord EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_norecord,-20061);

CURSOR cust\_trans IS

SELECT po.owner\_id, po.owner\_name, po.owner\_contact, po.owner\_gender, po.state, COUNT(t.transaction\_dateTime) AS TotalTransaction, MAX(t.transaction\_dateTime) AS LastTransaction

FROM petowner po, transaction t

WHERE t.owner\_id=po.owner\_id AND EXTRACT(YEAR FROM t.transaction\_dateTime) = IN\_year

GROUP BY po.owner\_id, po.owner\_name, po.owner\_contact, po.owner\_gender, po.state;

BEGIN

counter := 0;

record\_count := 0;

SELECT COUNT(\*) INTO record\_count

FROM petowner po, transaction t

WHERE t.owner\_id=po.owner\_id AND EXTRACT(YEAR FROM t.transaction\_dateTime) = IN\_year;

IF record\_count = 0 THEN

RAISE\_APPLICATION\_ERROR(-20061,'No record found');

END IF;

DBMS\_OUTPUT.PUT\_LINE (chr(10));

DBMS\_OUTPUT.PUT\_LINE ('List of customer who do transaction for less than 3 times in the year '||IN\_year);

DBMS\_OUTPUT.PUT\_LINE('Report generated on : ' || TO\_CHAR(CURRENT\_DATE, 'DD-MM-YYYY HH:MI:SS') || ' by ' || USER);

DBMS\_OUTPUT.PUT\_LINE (chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 137, '-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Customer ID', 15, ' ') || RPAD('Customer Name', 26, ' ') || RPAD('Customer Contact', 20, ' ')|| RPAD('Gender', 10, ' ') || RPAD('State', 20, ' ') || RPAD('No of Transaction made', 25, ' ') || RPAD('Last Transaction Date', 30, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 137, '-'));

FOR cust\_record IN cust\_trans LOOP

IF cust\_record.totaltransaction < 3 THEN

DBMS\_OUTPUT.PUT\_LINE(RPAD(cust\_record.owner\_id, 15, ' ') || RPAD(cust\_record.owner\_name, 26, ' ') || RPAD(cust\_record.owner\_contact, 20, ' ')|| RPAD(cust\_record.owner\_gender, 10, ' ') || RPAD(cust\_record.state, 20, ' ') || RPAD(cust\_record.totaltransaction, 25, ' ') || RPAD(cust\_record.lasttransaction, 30, ' '));

counter := counter + 1;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 137, '='));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*',110,' ')||'Total No of Customer: '||counter);

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 137, '='));

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE ('No Veterinarian found');

WHEN e\_norecord THEN

DBMS\_OUTPUT.PUT\_LINE('--------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Failed to print report for ' || IN\_year || '.');

DBMS\_OUTPUT.PUT\_LINE('--------------------------------');

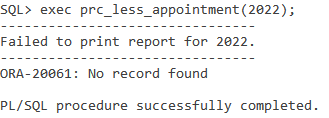
DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

exec prc\_less\_appointment(2020);

**Sample Output:**

****

****

**4.2 (Tan Teoh Xin Ee)**

**4.2.1 Query 1:Top Medicine Used in each branch (Strategic)**

**Purpose: The purpose of this query is to analyze the most used medicine in each branch. Therefore, we are able to tackle each and every branch using this query. For example, this medicine has the highest use in this branch, therefore the medicine is in demand in this area which we will have to focus the growth and stock in more of that kind of medicine to that branch.**

SQL statement:

clear break

clear compute

set linesize 80

set pagesize 100

break on state on branch\_id skip 1

compute SUM Label TOTAL of quantity percentage amount on branch\_id

TTITLE ON

TTITLE CENTER 'Top Medicine Used in each branch' SKIP 1-

CENTER ================================ SKIP 2

column branch\_id format a10

column branch\_id heading 'Branch ID'

column state format a13

column medic\_name format a20

column medic\_name heading 'Medicine Name'

column quantity heading 'Medicine|Used'

column amount format 99999999.99

column amount heading 'Total|Amount|(RM)'

create or replace view medicalUsed As

select t.branch\_id, d.medic\_id, m.medic\_name, sum(d.line\_qty) as quantity, sum(d.line\_total) as amount

from transaction t, branch b, transactiondetail d, medicalsupply m

where b.branch\_id = t.branch\_id

and t.transaction\_id = d.transaction\_id

and d.medic\_id = m.medic\_id

group by t.branch\_id, d.medic\_id,m.medic\_name

order by t.branch\_id,sum(d.line\_qty) DESC;

select u.branch\_id, b.state, u.medic\_name, u.quantity, u.amount,

RANK() over(partition by u.branch\_id order by u.quantity DESC) Ranks

from medicalused u, transaction t, branch b

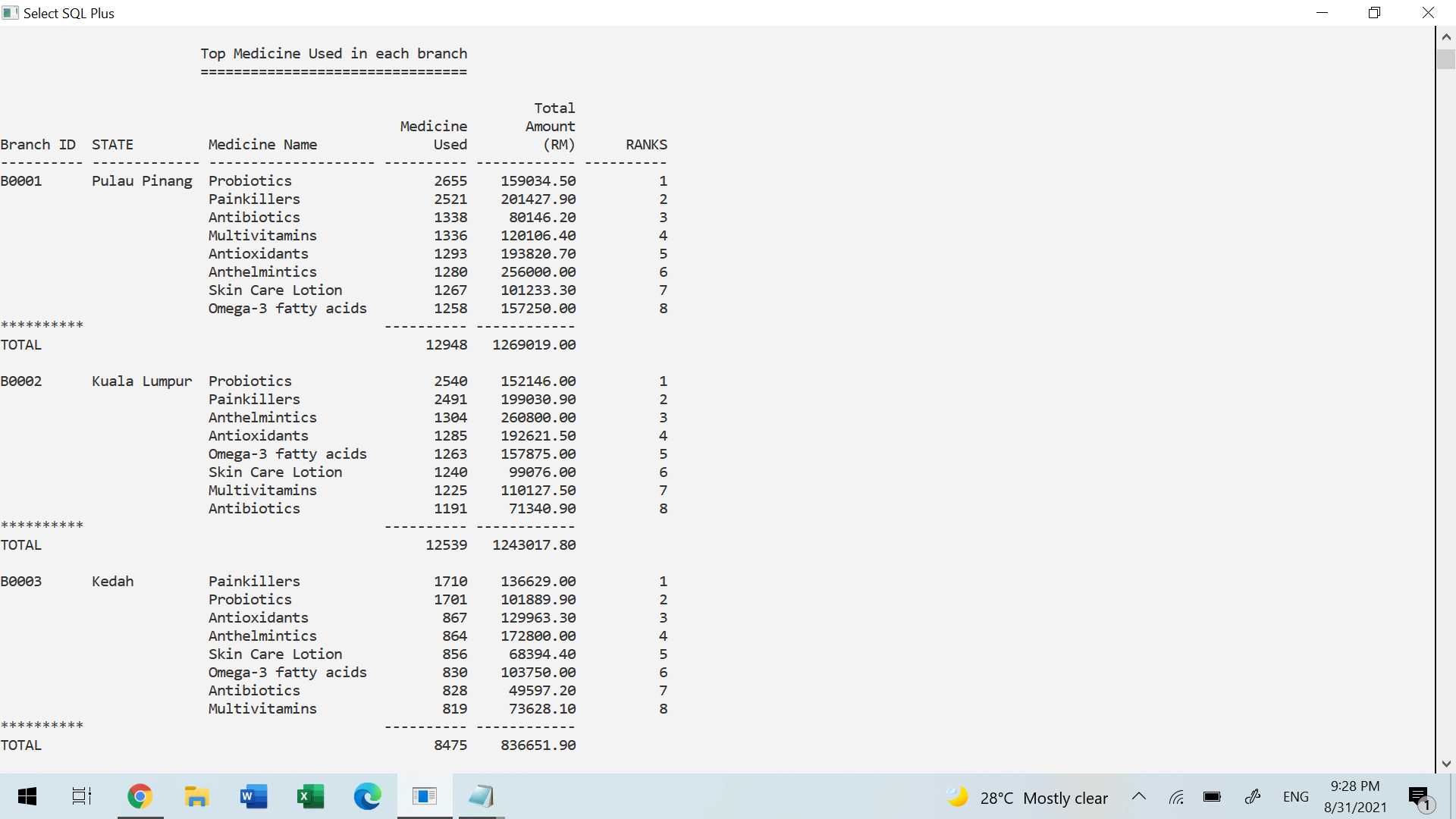
where u.branch\_id = t.branch\_id

and u.branch\_id = b.branch\_id

group by u.branch\_id, b.state, u.medic\_name, u.quantity, u.amount

order by u.branch\_id,u.quantity DESC;

**Sample Output:**



**4.2.2 Query 2: Top Veterinarian in each branch (Tactical)**

**Purpose: The purpose of this query is to know the most appointments received by each veterinarian in every branch. Therefore, we can revise the staffing levels by looking at their performance. For example, the most appointed veterinarian can be promoted to senior veterinarian.**

SQL statement:

clear break

clear compute

set linesize 80

set pagesize 100

break on state on branch\_id skip 1

compute SUM Label TOTAL of noofapp on branch\_id

TTITLE ON

TTITLE CENTER 'Top Veterinarian in each branch' SKIP 1-

CENTER =============================== SKIP 2

column branch\_id format a10

column branch\_id heading 'Branch ID'

column state format a13

column vet\_id format a6

column vet\_id heading 'Vet ID'

column vet\_name format a15

column vet\_name heading 'Vet Name'

column noofapp heading 'Appointment|Received'

create or replace view appointNum As

select count(appointment\_id) as NoOfapp, vet\_id

from appointment

group by vet\_id

order by count(appointment\_id) desc;

select b.branch\_id, b.state, v.vet\_name, a.vet\_id, a.noofapp,

RANK() over(partition by b.branch\_id order by a.noofapp DESC) RANKS

from branch b, veterinarian v, appointNum a

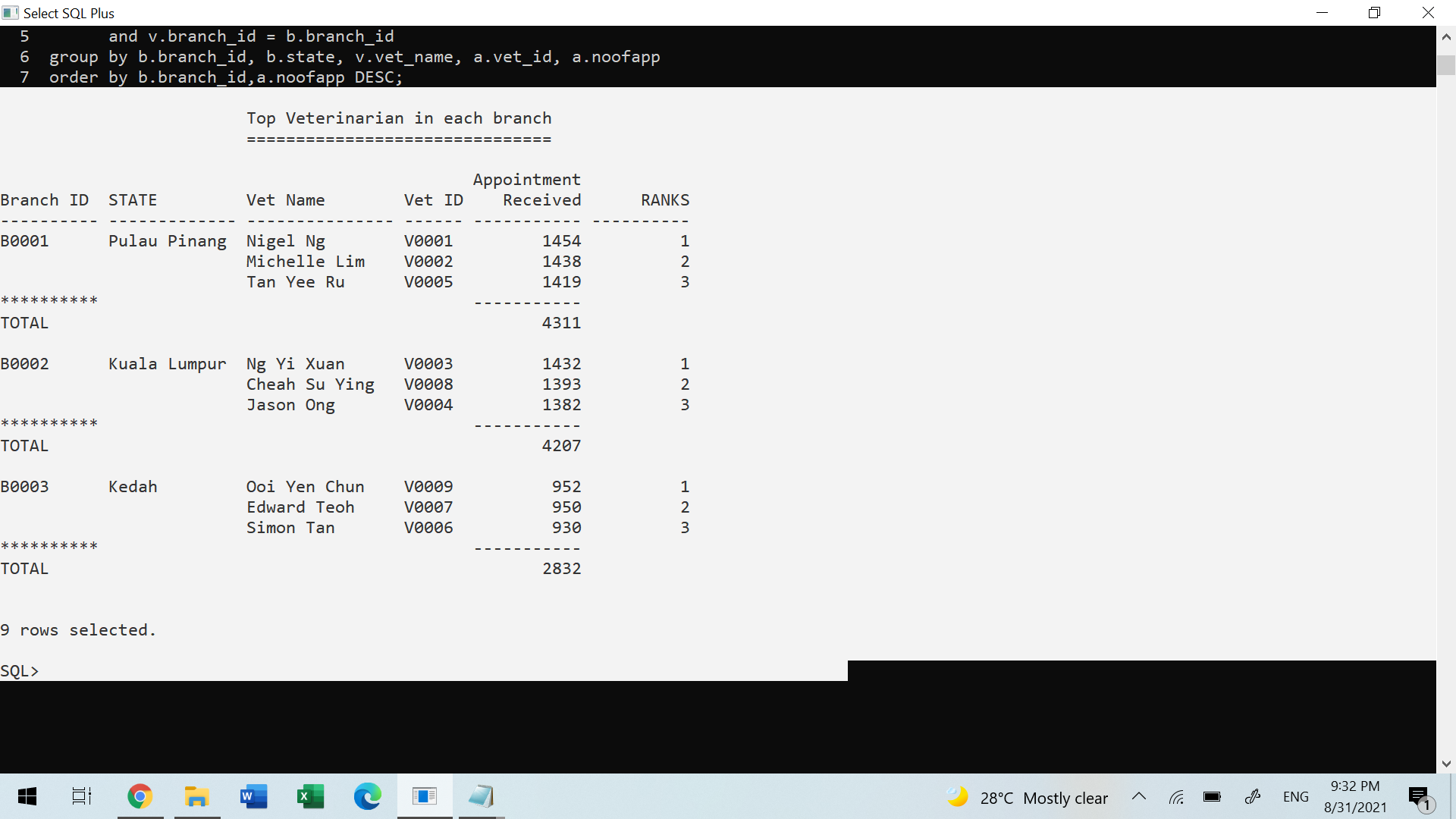
where a.vet\_id = v.vet\_id

and v.branch\_id = b.branch\_id

group by b.branch\_id, b.state, v.vet\_name, a.vet\_id, a.noofapp

order by b.branch\_id,a.noofapp DESC;

**Sample Output:**

****

**4.2.3 Query 3: Late Supplier List (Operational)**

**Purpose: The purpose of this query is to know the most frequent late supplier. By using this query, we are able to manage our stock such as the correct order/stock delay. This query is normally used by the low level staff in the shop.**

SQL statement:

clear break

clear compute

set linesize 100

set pagesize 150

BREAK ON REPORT

break on supplier\_contact on supplier\_name skip 1

compute Count Label NO.LATE of duration on supplier\_name

TTITLE ON

TTITLE CENTER 'Late Supplier list' SKIP 1-

CENTER ================== SKIP 2

column supplier\_name format a27

column supplier\_contact format a10

column supplier\_contact heading 'Contact No'

column purchase\_id format a11

column purchase\_id heading 'Purchase ID'

column purchase\_date format a9

column purchase\_date heading 'Purchase|Date'

column receive\_date format a9

column receive\_date heading 'Receive|Date'

column duration heading 'Duration|(Day)'

create or replace view difdate as

select purchase\_id, supplier\_id, purchase\_date, receive\_date,(receive\_date-purchase\_date) as duration

from purchaseTransaction

where receive\_date-purchase\_date>6

group by purchase\_id,supplier\_id, purchase\_date, receive\_date

order by supplier\_id;

select s.supplier\_name, s.supplier\_contact,d.purchase\_id, d.purchase\_date, d.receive\_date,d.duration

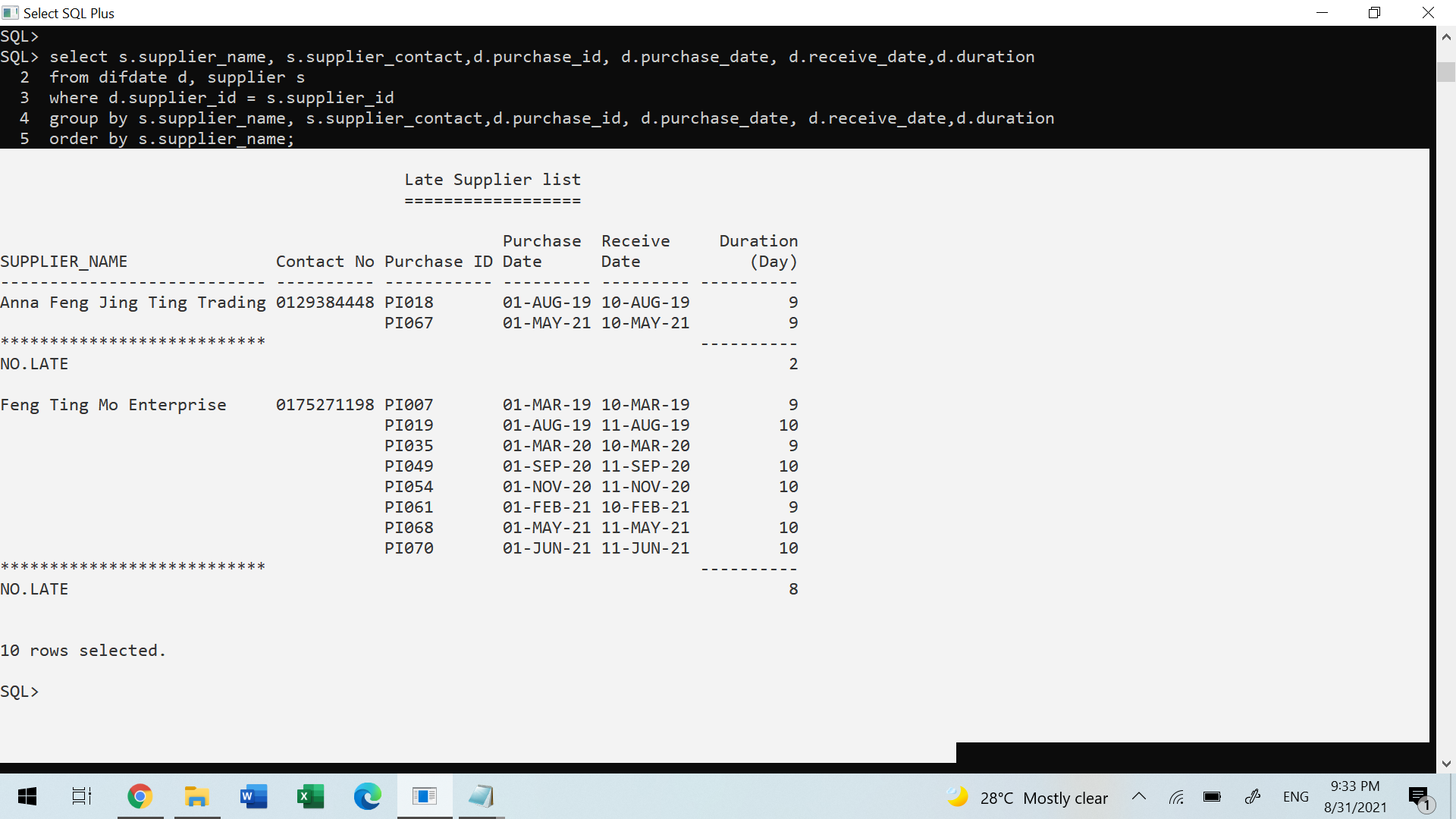
from difdate d, supplier s

where d.supplier\_id = s.supplier\_id

group by s.supplier\_name, s.supplier\_contact,d.purchase\_id, d.purchase\_date, d.receive\_date,d.duration

order by s.supplier\_name;

**Sample Output:**



**4.2.4 Procedure 1: Add Medical Supply**

**Purpose: The purpose of this procedure is to add new medicine into our database. If the user wishes to add new medicine, he/she can just call this procedure with two parameters(medicine name & price). Then the new record will auto be generated.**

SQL statement:

DROP SEQUENCE MEDICID;

CREATE SEQUENCE MEDICID

MINVALUE 8

MAXVALUE 9999

START WITH 9

INCREMENT BY 1;

CREATE OR REPLACE PROCEDURE PRC\_MEDICADD (IN\_MEDICNAME IN VARCHAR, IN\_MEDICPRICE IN NUMBER) As

v\_insertID char(5);

NEXT NUMBER;

BEGIN

NEXT:=MEDICID.NEXTVAL;

IF (NEXT<10) THEN

V\_INSERTID := TO\_CHAR('M000'||NEXT);

ELSIF (NEXT<100) THEN

V\_INSERTID := TO\_CHAR('M00'||NEXT);

ELSIF (NEXT<1000) THEN

V\_INSERTID := TO\_CHAR('M0'||NEXT);

ELSIF (NEXT<10000) THEN

V\_INSERTID := TO\_CHAR('M'||NEXT);

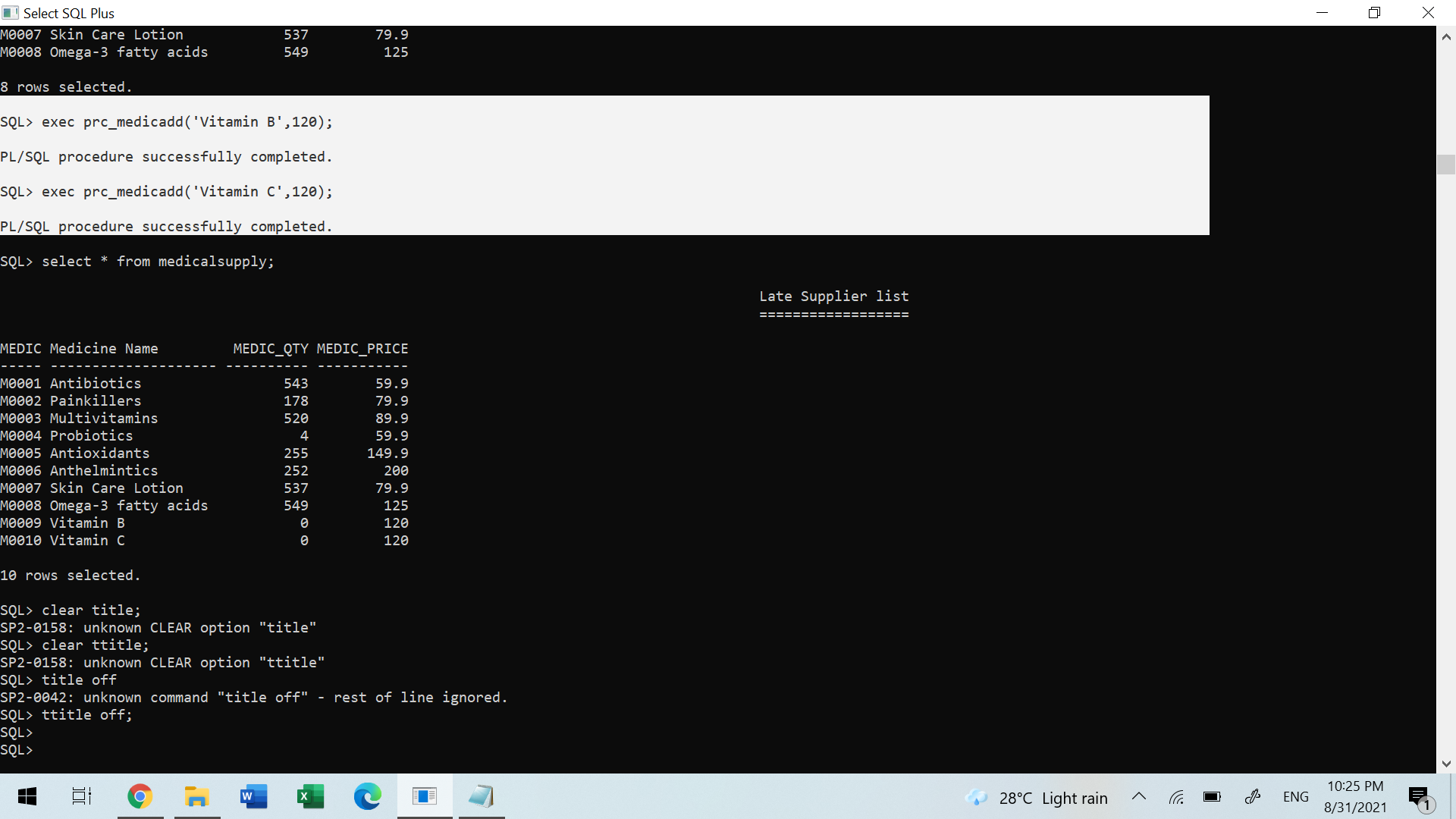
END IF;

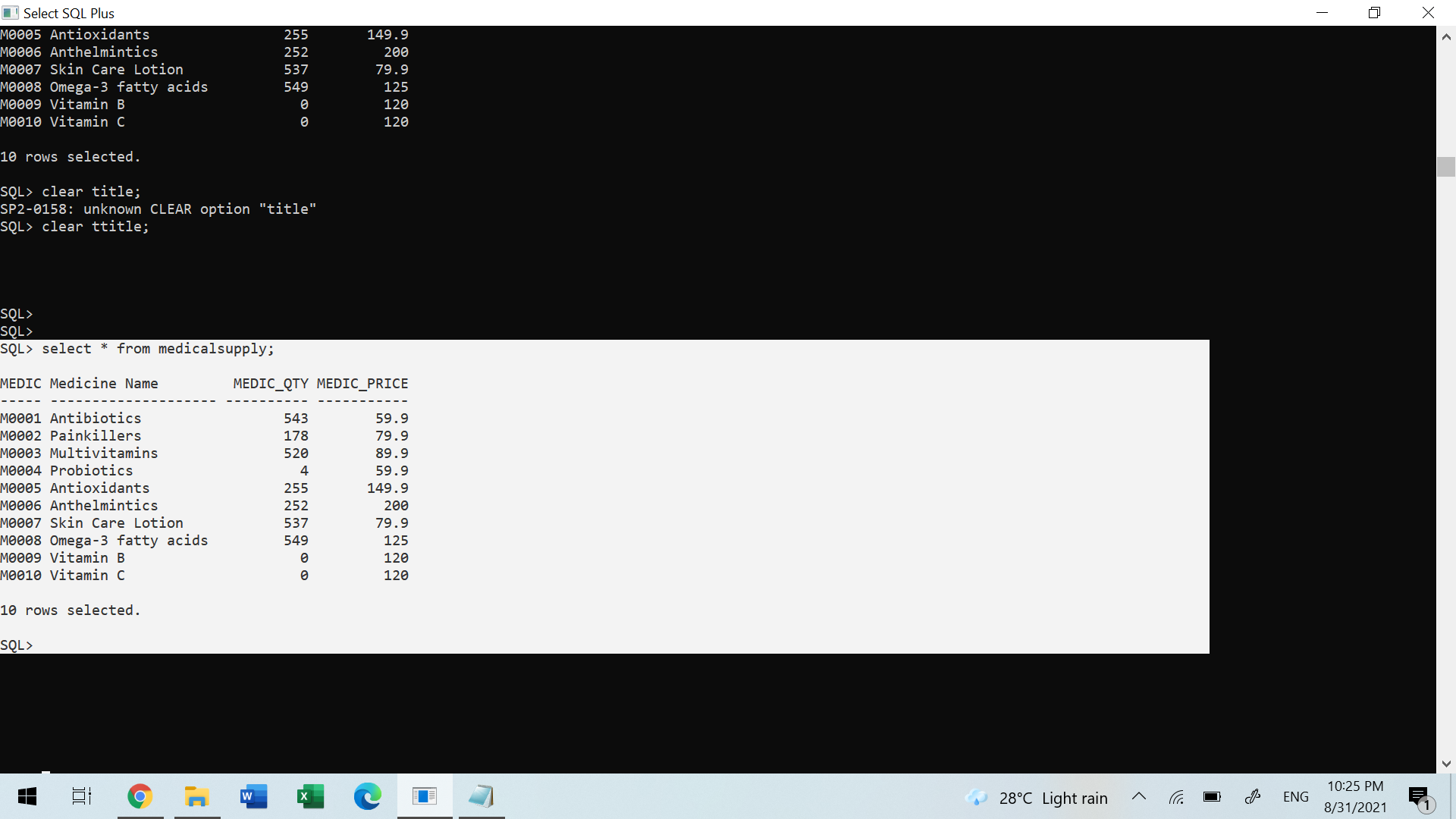
INSERT INTO MEDICALSUPPLY VALUES(V\_INSERTID,IN\_MEDICNAME,0,IN\_MEDICPRICE);

END;

/

**Sample Output:**

****

****

**4.2.5 Procedure 2: Delete Medical Supply**

**Purpose: The purpose of this procedure is to delete unwanted medicine from our database. If the user wishes to delete the medicine, he/she can just call this procedure with one parameter(medicine id). Then the specific record will be deleted permanently.**

SQL statement:

CREATE OR REPLACE PROCEDURE PRC\_MEDICDELETE(IN\_MEDICID IN CHAR) AS

BEGIN

DELETE FROM medicalsupply

WHERE medic\_id = IN\_MEDICID;

Exception

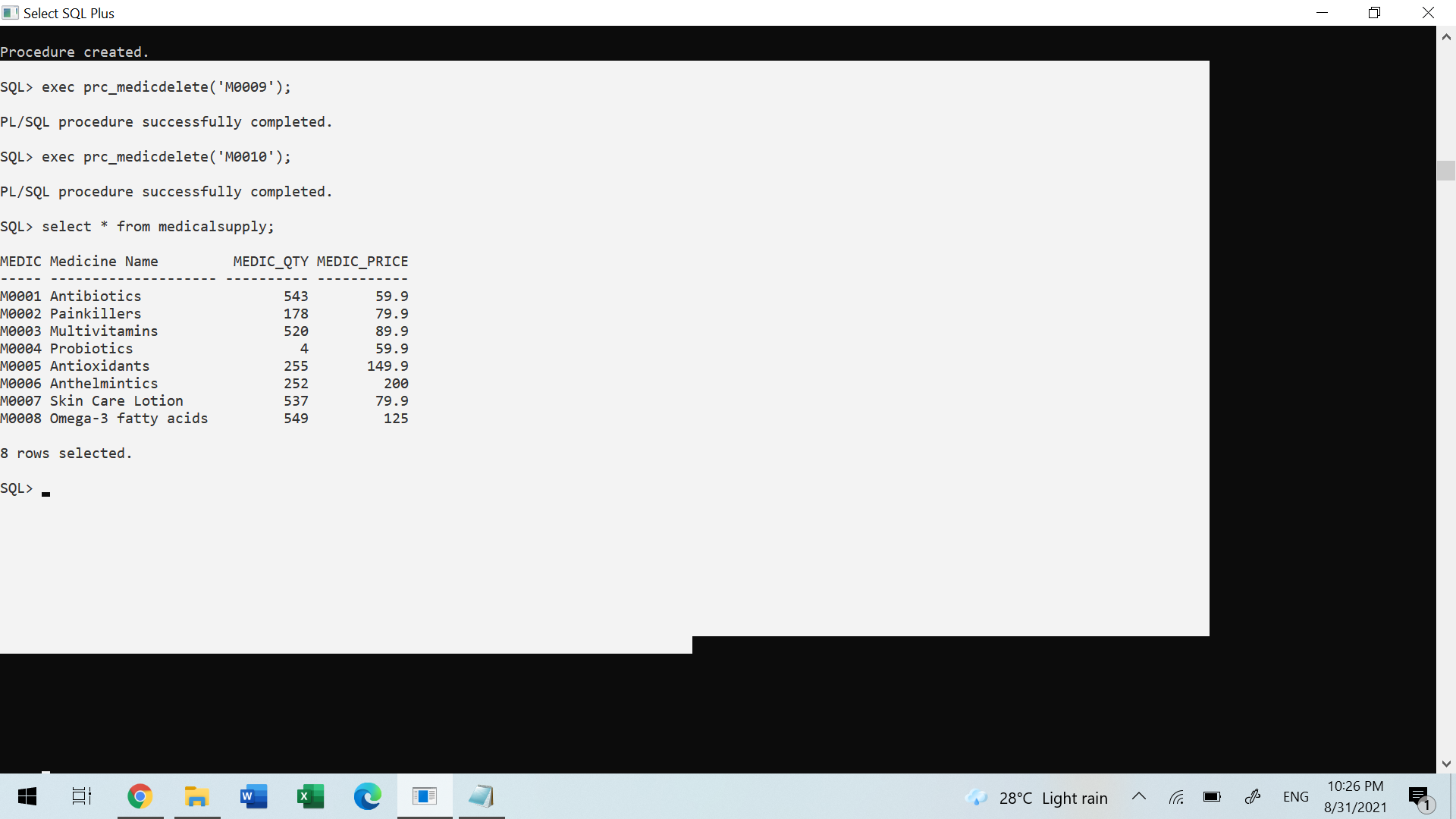
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NO SUCH MEDICINE!!!');

END;

/

**Sample Output:**

****

**4.2.6 Trigger 1: Add Stock Quantity (Add after purchasing stock)**

**Purpose: The purpose of this trigger is used to trigger after purchasing stock. For example, it will automatically update and add specific quantities of stock, after the company has purchased stock.**

SQL statement:

CREATE OR REPLACE TRIGGER TRG\_Add\_Stock\_Quantity

After Insert ON PurchaseItem

FOR EACH ROW

BEGIN

Update MedicalSupply

SET medic\_qty = medic\_qty + :new.purchase\_qty

where medic\_id = :new.medic\_id;

END;

/

**4.2.7 Trigger 2: Minus Stock Quantity (Minus after adding transaction detail)**

**Purpose: The purpose of this trigger is used to trigger after transaction detail. For example, it will automatically update and delete specific quantities of stock, after every transaction is made by the customer.**

SQL statement:

CREATE OR REPLACE TRIGGER TRG\_Minus\_Stock\_Quantity

After Insert ON TransactionDetail

FOR EACH ROW

BEGIN

Update MedicalSupply

SET medic\_qty = medic\_qty - :new.line\_qty

where medic\_id = :new.medic\_id;

END;

/

**4.2.8 Report 1: Detail report of Veterinarian Performance**

**Purpose: The purpose of this report is to know every veterinarian's performance in their branch such as how many transactions they have made and how much profit they bring toward the company. At the end, we are also able to know the grand total made by these three branches.**

SQL statement:

CREATE OR REPLACE PROCEDURE PRC\_VET\_PERFORMANCE IS

CURSOR BRANCH\_CURSOR IS

SELECT DISTINCT b.state

FROM veterinarian v, branch b

where v.branch\_id = b.branch\_id

ORDER BY b.state desc;

cursor vet\_cursor (branches in char)is

select v.vet\_id, b.state, v.vet\_name, v.vet\_contact, v.vet\_gender, count(t.transaction\_id) as transaction, sum(t.total\_amount)as amount

from veterinarian v, appointment a, transaction t, branch b

where v.vet\_id= a.vet\_id

and v.branch\_id = b.branch\_id

and b.state = branches

and a.appointment\_id = t.appointment\_id

group by v.vet\_id, b.state, v.vet\_name, v.vet\_contact, v.vet\_gender

order by v.vet\_id;

v\_totalAmt NUMBER(17,2) := 0;

v\_totalTrans NUMBER(10) := 0;

v\_grandTotal NUMBER(17,2) := 0;

v\_grandTrans NUMBER(10) := 0;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 48, ' ') || 'Report generated on : ' || TO\_CHAR(CURRENT\_DATE, 'DD-MM-YYYY HH:MI:SS') || 'by ' || USER);

DBMS\_OUTPUT.PUT\_LINE(chr(10));

FOR branches IN BRANCH\_CURSOR LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD('Branch', 20, ' ') || ': ' ||RPAD(UPPER(branches.state), 60, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 95, '-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Staff ID', 10, ' ') || RPAD('StaffName', 25, ' ') || RPAD('Staff Tel', 15, ' ') || RPAD('Gender', 10, ' ')|| RPAD('Total Transactions', 20, ' ') || RPAD('Total Amount', 17, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 95, '-'));

v\_totalTrans := 0;

v\_totalAmt := 0;

FOR vet\_rec IN vet\_CURSOR(branches.state) LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD(vet\_rec.vet\_id, 10, ' ') ||RPAD(vet\_rec.vet\_name, 25, ' ') || RPAD(vet\_rec.vet\_contact, 15, ' ') || RPAD(vet\_rec.vet\_gender, 10, ' ')|| RPAD(vet\_rec.transaction, 20, ' ') || 'RM ' ||RPAD(TRIM(TO\_CHAR(vet\_rec.amount, '999G999G999D99')), 20, ' '));

v\_totalTrans := v\_totalTrans + vet\_rec.transaction;

v\_totalAmt := v\_totalAmt + vet\_rec.amount;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 95, '-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 51, ' ') || 'Subtotal:' ||RPAD(v\_totalTrans, 4, ' ') || RPAD(' ', 16, ' ') || 'RM ' ||RPAD(TRIM(TO\_CHAR(v\_totalAmt, '9G999G999G999D99')), 20, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 95, '-'));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

v\_grandTrans := v\_grandTrans + v\_totalTrans;

v\_grandTotal := v\_grandTotal + v\_totalAmt;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 95, '-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 48, ' ') || 'Grand Total:' ||RPAD(v\_grandTrans, 5, ' ') || RPAD(' ', 15, ' ') || 'RM ' ||RPAD(TRIM(TO\_CHAR(v\_grandTotal, '9G999G999G999D99')), 25, ' ') || RPAD(' ', 3, ' '));

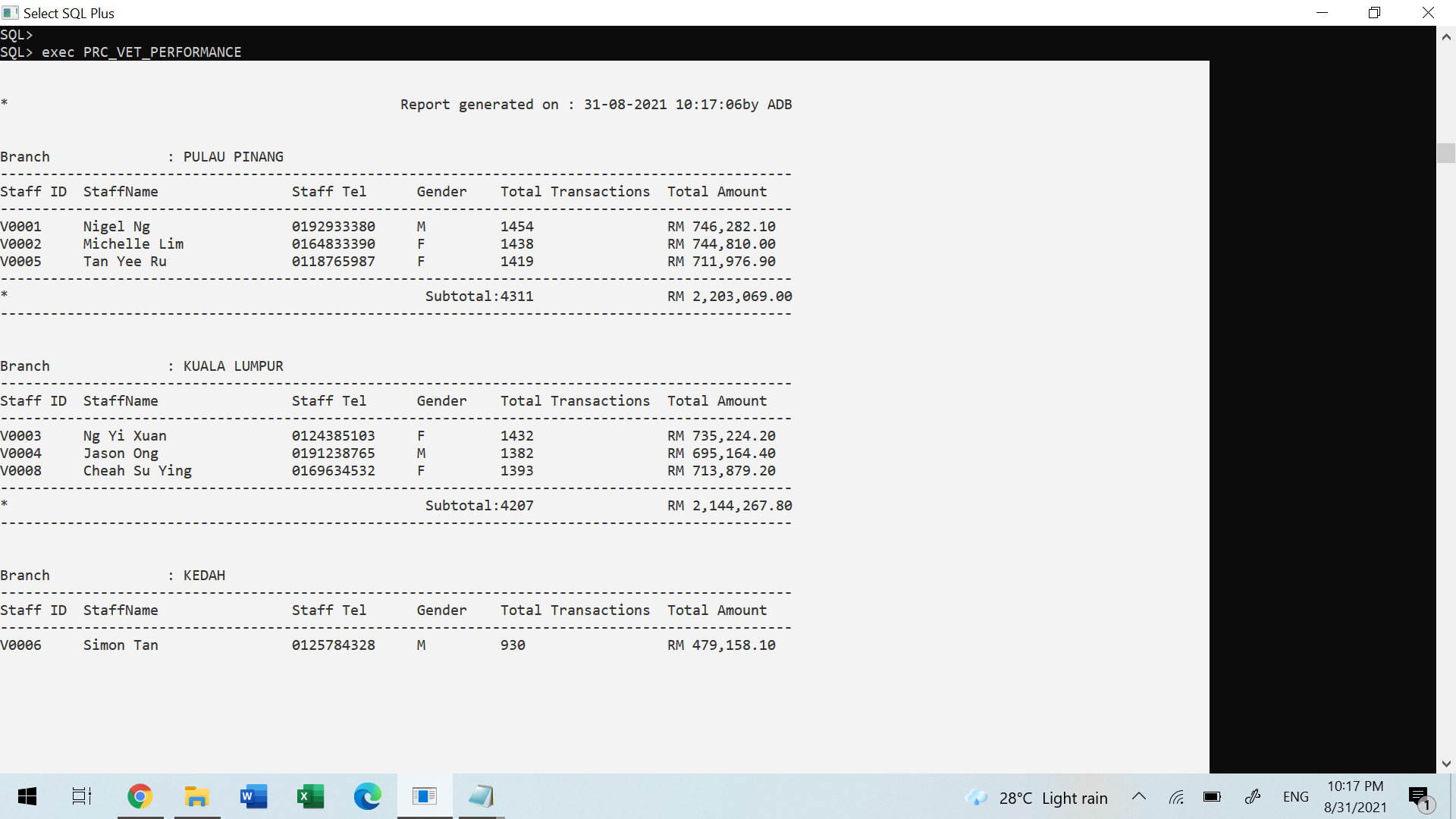
DBMS\_OUTPUT.PUT\_LINE(chr(10));

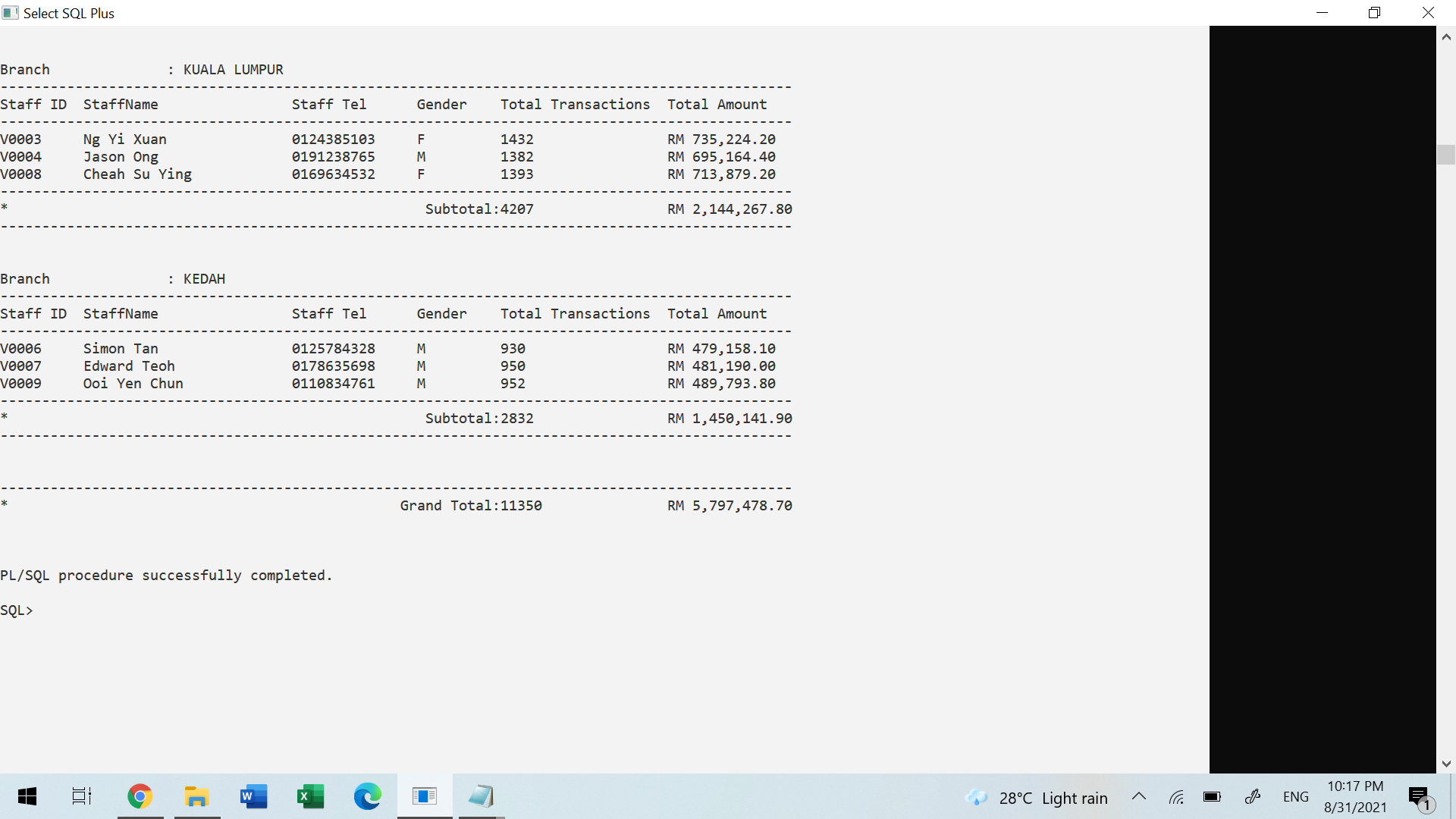
END;

/

exec PRC\_VET\_PERFORMANCE

**Sample Output:**

****

****

**4.2.9 Report 2: Summary report of Supplier Profile**

**Purpose: The purpose of this report is to summarize the supplier information such as the company details, what product they supplied to us and the purchase transaction details with them. At the end, we can know the total transaction made and total stock we get supplied from the specific supplier.**

SQL statement:

ALTER SESSION SET NLS\_DATE\_FORMAT = 'DD-MM-YYYY';

create or replace procedure prc\_supplier\_report(suppliercode in CHAR) Is

cursor purchaseTransaction\_cursor Is

select p.purchase\_id, p.purchase\_date, p.receive\_date, p.purchase\_amount

from purchaseTransaction p, supplier s

where s.supplier\_id = p.supplier\_id

and p.supplier\_id = suppliercode;

cursor purchaseItem\_cursor Is

select m.medic\_id, m.medic\_name, sum(p.purchase\_qty) as quantity

from medicalsupply m, purchaseItem p, purchaseTransaction t

where m.medic\_id = p.medic\_id

and t.purchase\_id = p.purchase\_id

and t.supplier\_id = suppliercode

group by m.medic\_id, m.medic\_name

order by m.medic\_id;

EXCE\_SUPPLIERCODE EXCEPTION;

PRAGMA EXCEPTION\_INIT(EXCE\_SUPPLIERCODE, -20310);

totalorder number(11,2);

totalitem number(12,2):=0;

totalqty number(12,2):=0;

v\_supplierID CHAR(6);

v\_supplierName VARCHAR2(30);

v\_suppliercontact NUMBER(15):=0;

V\_VALIDSUPPLIERID CHAR(6);

V\_prodrate NUMBER(5,2);

BEGIN

V\_VALIDSUPPLIERID := SUPPLIERCODE;

IF V\_VALIDSUPPLIERID = ' ' THEN

RAISE EXCE\_SUPPLIERCODE;

ELSE

DBMS\_OUTPUT.put\_line(chr(10));

DBMS\_OUTPUT.PUT\_LINE('SUPPLIER SUMMARY REPORT FOR '||UPPER(SUPPLIERCODE));

DBMS\_OUTPUT.PUT\_LINE('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

SELECT S.supplier\_id, supplier\_name, supplier\_contact INTO v\_supplierid,v\_suppliername,v\_suppliercontact

FROM supplier S

WHERE S.supplier\_id = suppliercode;

SELECT sum(purchase\_qty) as TotalQty INTO TotalItem

FROM purchaseItem;

DBMS\_OUTPUT.put\_line(rpad('Supplier ID',20,' ')||':'||v\_supplierid);

DBMS\_OUTPUT.put\_line(rpad('Supplier Name',20,' ')|| ':'||v\_supplierName);

DBMS\_OUTPUT.put\_line(rpad('Supplier Contact',20,' ')|| ':'||v\_suppliercontact);

DBMS\_OUTPUT.put\_line(chr(10));

totalorder := 0;

DBMS\_OUTPUT.PUT\_LINE('Past Supplied Record:');

DBMS\_OUTPUT.PUT\_LINE('|'||rpad('-',33,'-')||'|');

DBMS\_OUTPUT.PUT\_LINE('|'||rpad('Purchase ID',15,' ') ||' | '||rpad('Supplied Date',15,' ')||'|');

DBMS\_OUTPUT.PUT\_LINE('|'||rpad('-',33,'-')||'|');

for purch\_rec in purchaseTransaction\_cursor loop

IF purch\_rec.receive\_date IS NULL THEN

DBMS\_OUTPUT.PUT\_LINE('|'||rpad(purch\_rec.purchase\_id,15,' ') || ' | '||rpad('-',15,' ')||'|');

ELSE

DBMS\_OUTPUT.PUT\_LINE('|'||rpad(purch\_rec.purchase\_id,15,' ') || ' | '||rpad(purch\_rec.receive\_date,15,' ')||'|');

totalorder := totalorder + 1;

END IF;

END LOOP;

totalqty := 0;

DBMS\_OUTPUT.PUT\_LINE('|'||rpad('-',33,'-')||'|');

DBMS\_OUTPUT.put\_line(chr(10));

DBMS\_OUTPUT.PUT\_LINE('Top Items Supplied Percentage:');

DBMS\_OUTPUT.PUT\_LINE(rpad('-',54,'-'));

for purch\_item in purchaseItem\_cursor loop

V\_prodrate:=0;

V\_prodrate:= (purch\_item.quantity/totalitem)\*100;

DBMS\_OUTPUT.PUT\_LINE(rpad(purch\_item.medic\_id,8,' ') ||' \* '|| rpad(purch\_item.medic\_name,30,' ')||' \* ' ||lpad(V\_prodrate,8,' ')||'%');

DBMS\_OUTPUT.PUT\_LINE(rpad('-',54,'-'));

totalqty := totalqty + purch\_item.quantity;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE('Total Orders Made :' ||totalorder);

DBMS\_OUTPUT.PUT\_LINE('Total Items Supplied :' ||totalqty);

END IF;

EXCEPTION

WHEN EXCE\_SUPPLIERCODE THEN

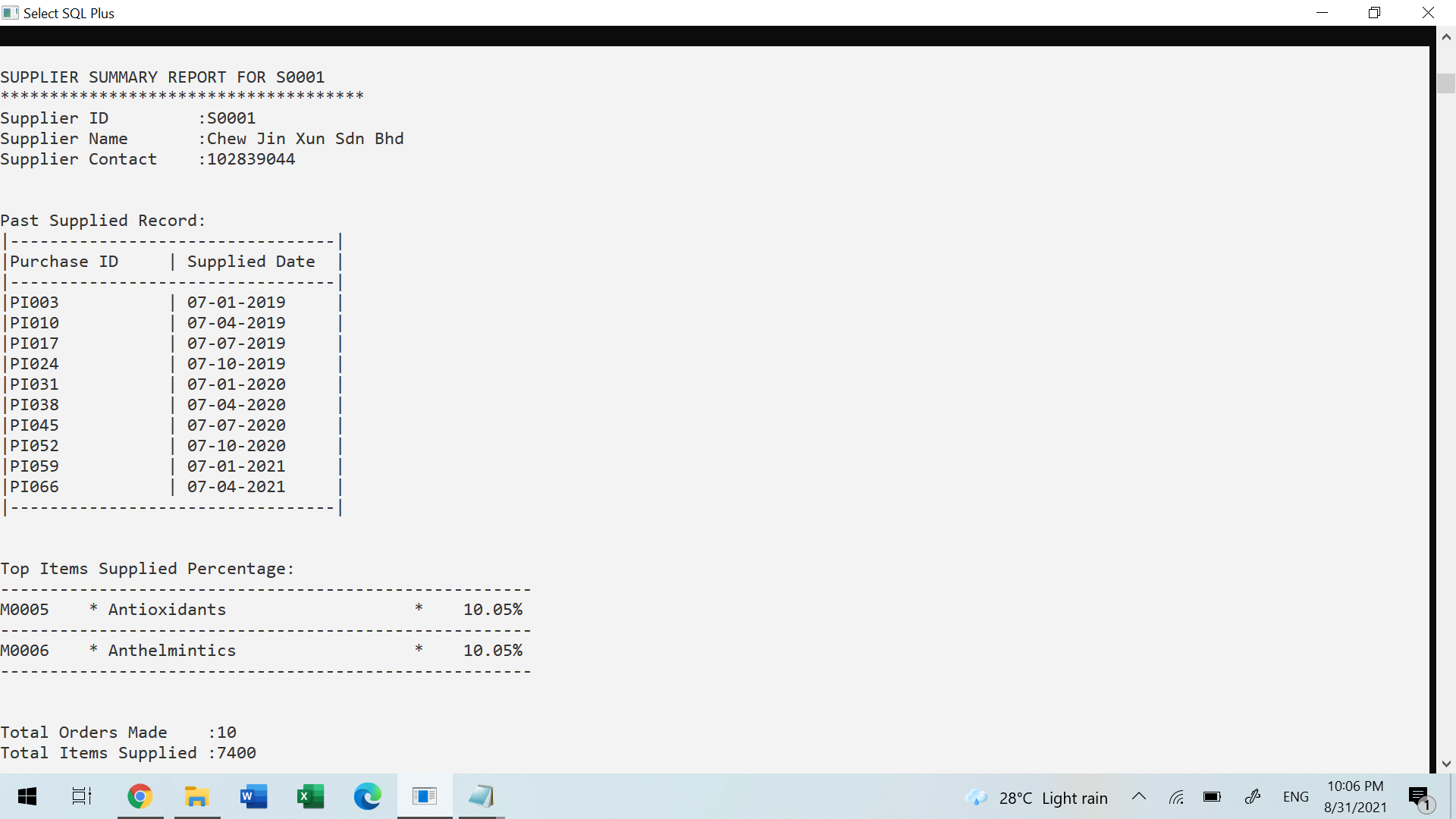
DBMS\_OUTPUT.PUT\_LINE(-20310||'INVALID SUPPLIER CODE.');

END;

/

exec PRC\_SUPPLIER\_REPORT('S0001');

**Sample Output:**

****

**4.2.10 Report 3: On demand report of Inventory value**

**Purpose: The purpose of this report is to get the detail of our company inventory value such as how many stocks we left for every medicine and how much does it cost, which means the property company has in the warehouse. At the end, we can know the total amount of inventory and cost.**

SQL statement:

set pagesize 1000

set linesize 200

create or replace procedure PRC\_INVENTORY\_REPORT is

cursor medic\_cursor Is

select medic\_id, medic\_name, medic\_qty, medic\_price, (medic\_qty\*medic\_price) as amount

from medicalsupply

group by medic\_id, medic\_name, medic\_qty, medic\_price

order by medic\_id;

medic\_c medic\_cursor% ROWTYPE;

subtotal NUMBER;

totalqty NUMBER;

tunitcost NUMBER;

tproduct NUMBER;

grandqty NUMBER;

grandtotal NUMBER;

BEGIN

DBMS\_OUTPUT.put\_line(chr(10));

DBMS\_OUTPUT.PUT\_LINE(rpad(chr(9),7,chr(9))||rpad('Date',17,' ')||':'|| SYSDATE);

DBMS\_OUTPUT.PUT\_LINE(rpad(chr(9),7,chr(9))||rpad('Time',17,' ')||':'||TO\_CHAR(SYSDATE,'HH24:MI:SS'));

DBMS\_OUTPUT.PUT\_LINE(rpad(chr(9),7,chr(9))||rpad('Day',17,' ')||':'||TO\_CHAR(SYSDATE,'DAY'));

DBMS\_OUTPUT.PUT\_LINE(rpad(chr(9),7,chr(9))||rpad('Generated By',17,' ')||':' || USER);

DBMS\_OUTPUT.put\_line(chr(10));

DBMS\_OUTPUT.PUT\_LINE(' LATEST TOTAL INVENTORY VALUE REPORT');

DBMS\_OUTPUT.PUT\_LINE('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

DBMS\_OUTPUT.PUT\_LINE(rpad('-',7,'-')||' '||rpad('-',34,'-')||' '||rpad('-',11,'-')|| ' '||rpad('-',11,'-')||' '||rpad('-',17,'-'));

DBMS\_OUTPUT.PUT\_LINE(rpad('MedicID',8,' ')||rpad('Medic Name',35,' ')|| rpad('Stock Qty',12,' ')|| rpad('Unit Cost',13,' ')||rpad('Cost',15,' '));

DBMS\_OUTPUT.PUT\_LINE(rpad('-',7,'-')||' '||rpad('-',34,'-')||' '||rpad('-',11,'-')|| ' '||rpad('-',11,'-')||' '||rpad('-',17,'-'));

tproduct:=0;

Subtotal:=0;

totalqty:=0;

tunitcost:=0;

OPEN medic\_cursor;

LOOP

FETCH medic\_cursor INTO medic\_c;

EXIT WHEN medic\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(rpad(medic\_c.medic\_id,8,' ')||rpad(medic\_c.medic\_name,35,' ')|| rpad(medic\_c.medic\_qty,12,' ')||'RM'||lpad(TRIM(TO\_CHAR(medic\_c.medic\_price,'999G999D99')),9,' ')||' RM'||lpad( TRIM(TO\_CHAR(medic\_c.Amount,'999G999D99')),15,' '));

subtotal:= subtotal + medic\_c.Amount;

totalqty:=totalqty + medic\_c.medic\_qty;

tunitcost:=tunitcost+ medic\_c.medic\_price;

tproduct:= tproduct + 1;

END LOOP;

grandqty := totalqty;

grandtotal := subtotal;

DBMS\_OUTPUT.PUT\_LINE(rpad('-',7,'-')||' '||rpad('-',34,'-')||' '||rpad('-',11,'-')|| ' '||rpad('-',11,'-')||' '||rpad('-',17,'-'));

DBMS\_OUTPUT.PUT\_LINE(rpad('\*',43,' ')||rpad(totalqty,12,' ')||'RM'|| lpad(TRIM(TO\_CHAR(tunitcost,'999G999D99')),9,' ')||' RM'||lpad(TRIM(TO\_CHAR(subtotal,'999G999G999D99')),15,' '));

DBMS\_OUTPUT.PUT\_LINE(rpad('-',7,'-')||' '||rpad('-',34,'-')||' '||rpad('-',11,'-')|| ' '||rpad('-',11,'-')||' '||rpad('-',17,'-'));

DBMS\_OUTPUT.put\_line(chr(10));

DBMS\_OUTPUT.put\_line('SUMMARY');

DBMS\_OUTPUT.PUT\_LINE(rpad('Total Inventory Items Count ',40,' ')||':'||grandqty|| ' item(s)' );

DBMS\_OUTPUT.PUT\_LINE(rpad('Total Inventory Value ',40,' ')||':RM'||TRIM(TO\_CHAR(grandtotal,'999G999G999D99')) );

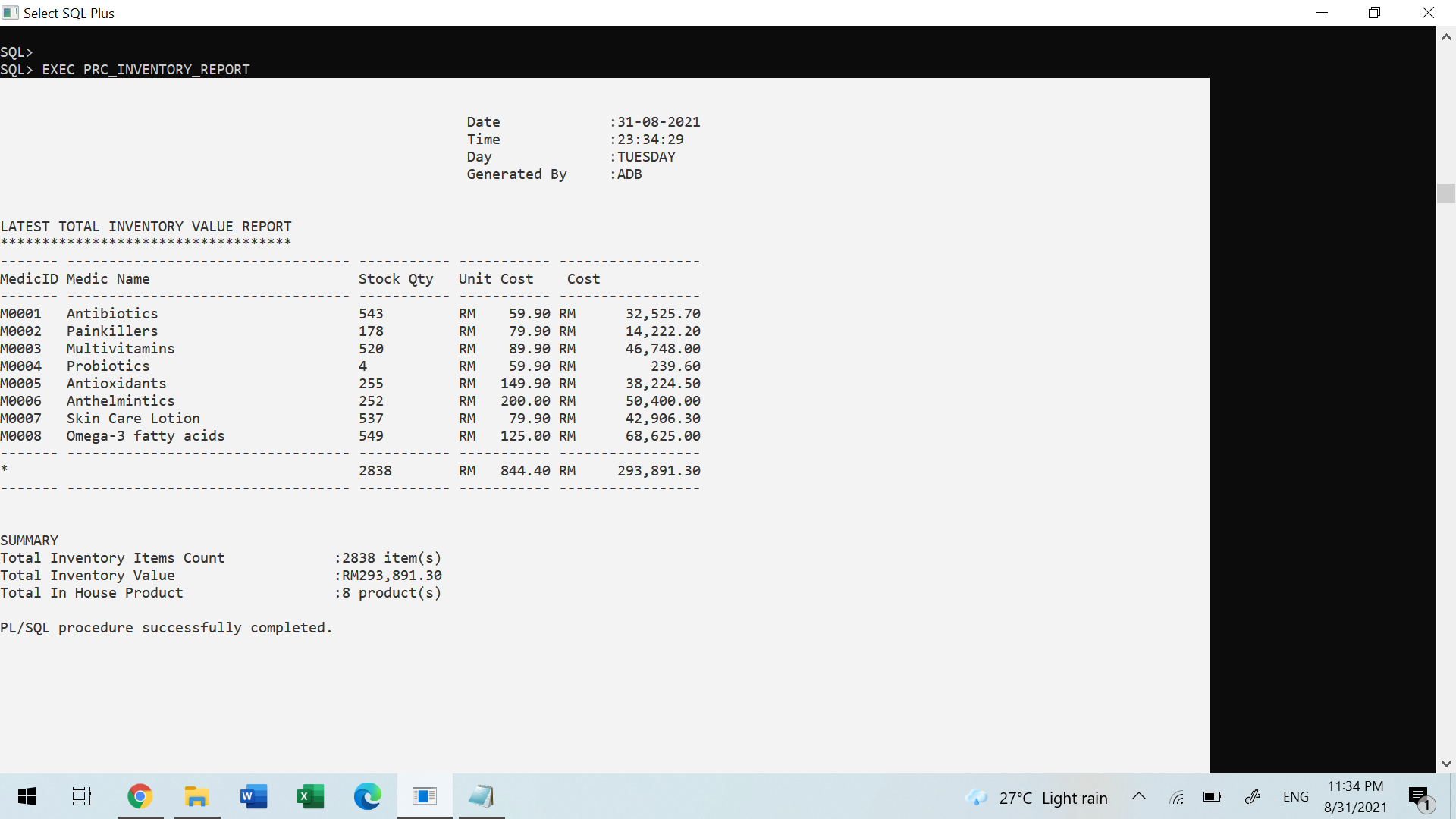
DBMS\_OUTPUT.PUT\_LINE(rpad('Total In House Product ',40,' ')||':'||tproduct|| ' product(s)' );

END;

/

EXEC PRC\_INVENTORY\_REPORT

**Sample Output:**

****

**4.3 (Tan Wei Siong)**

**4.3.1 Query 1: Top and Least Treatment Last Year for Each Branch**

**Purpose: The purpose of this query is to let the manager to view the top and least treatment in the last year for each branch. With the query, the manager can determine which treatment is not famous in the branch. Thus, the manager can make the pricing strategy to promote their least treatment to their customer. The manager can also know the quantity difference between the most and the least treatment .**

SET LINESIZE 150;

SET PAGESIZE 150;

COLUMN TREATMENT\_TYPE FORMAT A30;

TTITLE CENTER ('TOP AND LEAST TREATMENT LAST YEAR FOR EACH BRANCH') SKIP 2

WITH outlets AS

(SELECT b.branch\_id, t.treatment\_id, t.treatment\_type, COUNT(t.treatment\_id) AS Branch\_total\_qty

FROM Treatment t, Appointment a, Transaction v, Branch b

WHERE t.treatment\_id = a.treatment\_id AND

a.appointment\_id = v.appointment\_id AND

b.branch\_id = v.branch\_id AND

Extract(year from Appointment\_datetime) = Extract(year from sysdate)-1

GROUP by b.branch\_id, t.treatment\_id, t.treatment\_type

order by 1,4),

mn\_trans AS

(SELECT branch\_id, MIN(Branch\_total\_qty) AS Least

FROM outlets

group by branch\_id

ORDER BY 1),

mx\_trans AS

(SELECT branch\_id,MAX(Branch\_total\_qty) AS Top

FROM outlets

group by branch\_id

ORDER BY 1)

SELECT A.branch\_id,C.treatment\_id, C.treatment\_type, A.Least, D.treatment\_id,D.treatment\_type, B.Top

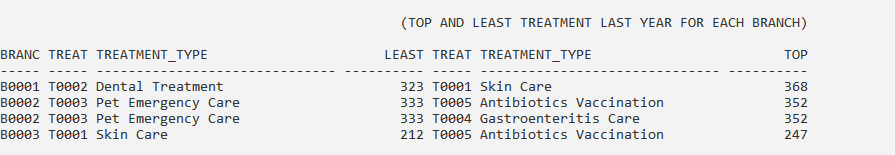
From mn\_trans A JOIN mx\_trans B ON (A.branch\_id = B.branch\_id)

JOIN outlets C ON (A.branch\_id = C.branch\_id) AND c.Branch\_total\_qty = A.Least

JOIN outlets D ON (A.branch\_id = D.branch\_id) AND D.Branch\_total\_qty = B.Top

ORDER BY 1;

**Sample Output:**



**4.3.2 Query 2: Selected Pet Type Age Group and Respective Visit Quantity**

**Purpose: The purpose of this query is to find the selected pet type and the total visit quantity. With this query, the management can determine the number of selected pet type age groups and the total visit for the age group. The management can find the most number of the age group for the pet but having the low number of visits and propose the strategy to attract the age group. For instance, if the newborn pet dog has a large quantity but low visit, the management can make a promotion plan on the suitable treatment to attract the potential customer.**

BREAK ON REPORT

TTITLE CENTER ('AGE GROUP PET AND TOTAL VISIT') SKIP 2

COMPUTE SUM OF Total\_Visit ON REPORT;

COMPUTE SUM OF Total\_Qty\_Age\_Group ON REPORT;

COLUMN type\_id HEADING 'TYPE|ID';

COLUMN type\_name HEADING 'TYPE|NAME';

COLUMN Total\_Qty\_Age\_Group HEADING 'TOTAL|QTY|AGE|GROUP';

COLUMN Total\_Visit HEADING 'TYPE|VISIT';

SET LINESIZE 80;

Accept life\_span prompt "Please enter Max Life Span for the pet: " default 10

Accept petType\_id prompt "Please enter PetType\_ID: " default 10

WITH ageGroup as(

SELECT CASE

WHEN (EXTRACT (YEAR FROM sysdate) - EXTRACT(YEAR FROM p.pet\_dob) > ROUND(&life\_span \* 2/3)) THEN 'Old'

WHEN (EXTRACT (YEAR FROM sysdate) - EXTRACT(YEAR FROM p.pet\_dob) > ROUND(&life\_span \* 1/3)) THEN 'Adult'

ELSE 'New Born'

END AS age\_group, p.pet\_id, p.type\_id, t.type\_name

FROM Pet p, PetType t

WHERE p.type\_id = t.type\_id

ORDER BY 1

),

visitTime as (SELECT b.type\_id, b.type\_name, b.age\_group,

COUNT(a.appointment\_dateTime) AS Total\_Visit

FROM ageGroup b, Appointment a

WHERE a.pet\_id = b.pet\_id

GROUP BY type\_id, type\_name, age\_group

),

totalQtyAgeGroup as (SELECT type\_id, type\_name, age\_group, COUNT(age\_group) As Total\_QTY\_Age\_Group

FROM ageGroup

GROUP BY type\_id, type\_name, age\_group

)

SELECT V.type\_id, v.type\_name, v.age\_group, t.Total\_Qty\_Age\_Group, v.Total\_Visit

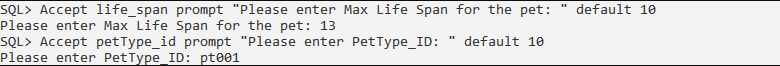
FROM visitTime V

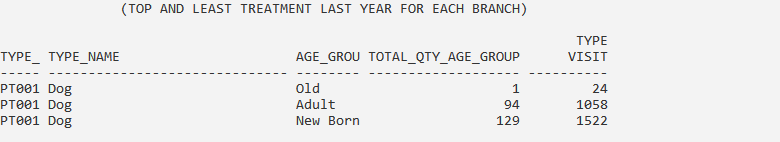
LEFT JOIN totalQtyAgeGroup t ON V.type\_id = t.type\_id AND t.age\_group = v.age\_group

WHERE V.type\_id = UPPER('&petType\_id')

ORDER BY 1,2;

**Sample Output:**





**4.3.3 Query 3: Gross Profit and Net Profit Previous Year**

**Purpose: The purpose of this query is to view the gross profit and net profit in the previous year. The profit will be listed in each month so that the management can know the performance of the pet clinic. The gross profit is calculated based on the transaction for all branches and the purchase amount is the sum of the purchase medicine stock price. The net profit will be calculated by gross profit - purchase amount.**

COLUMN GROSS\_PROFIT FORMAT '9,999,999.99';

COLUMN PURCHASE\_AMOUNT FORMAT '9,999,999.99';

COLUMN NET\_PROFIT FORMAT '9,999,999.99';

COLUMN GROSS\_PROFIT HEADING 'GROSS|PROFIT';

COLUMN PURCHASE\_AMOUNT HEADING 'PURCHASE|AMOUNT';

COLUMN NET\_PROFIT HEADING 'NET|PROFIT';

TTITLE CENTER ('GROSS AND NET PROFIT IN LAST YEAR') SKIP 2

SET LINESIZE 100;

SET PAGESIZE 200;

BREAK ON REPORT

COMPUTE SUM OF NET\_PROFIT ON REPORT

COMPUTE SUM OF NO\_Of\_Transaction ON REPORT;

COMPUTE SUM OF GROSS\_PROFIT ON REPORT;

COMPUTE SUM OF PURCHASE\_AMOUNT ON REPORT;

COMPUTE SUM OF NET\_PROFIT ON REPORT;

WITH last\_year\_transaction AS(

SELECT transaction\_id, transaction\_dateTime

FROM Transaction

WHERE Extract(year from(transaction\_dateTime)) = extract(year from sysdate)-1),

last\_year\_purchase AS(

SELECT purchase\_id , purchase\_Date, purchase\_Amount

FROM PurchaseTransaction

WHERE Extract(year from(purchase\_Date)) = Extract(year from sysdate)-1),

purchase\_detail AS(

SELECT EXTRACT(Month from v.purchase\_Date) AS Month\_NO,

TO\_CHAR(v.purchase\_Date,'MON') as Month,

COUNT(v.purchase\_id) AS NO\_Of\_Purchase,

SUM(v.purchase\_Amount) AS PURCHASE\_AMOUNT

FROM last\_year\_purchase v

GROUP BY EXTRACT (Month from v.purchase\_Date),

TO\_CHAR(v.purchase\_Date,'MON')

)

SELECT EXTRACT(Month from v.transaction\_dateTime) AS Month\_NO,

TO\_CHAR(v.transaction\_dateTime,'MON') as Month,

COUNT(v.transaction\_id) AS NO\_Of\_Transaction,

SUM(t.total\_amount) AS GROSS\_PROFIT,

l.purchase\_Amount AS PURCHASE\_AMOUNT,

SUM(t.total\_amount) - l.purchase\_Amount AS NET\_PROFIT

FROM last\_year\_transaction v, Transaction t

LEFT JOIN purchase\_detail l ON l.Month\_NO =

EXTRACT(Month from transaction\_dateTime)

WHERE v.transaction\_id = t.transaction\_id

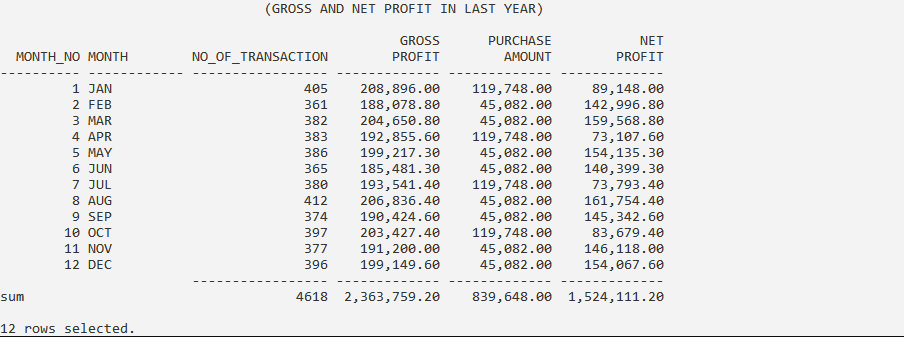
GROUP BY EXTRACT (Month from v.transaction\_dateTime),

TO\_CHAR(v.transaction\_dateTime,'MON'),

l.purchase\_Amount

ORDER BY 1;

**Sample Output:**



**4.3.4 Procedure 1: Pet Registration**

**Purpose: The purpose of this procedure is to let the staff register the pet in an easier way. The staff is only required to input the owner's phone number and their pet information into this procedure and the data will be stored in the database. If the owner contact is not found, an error message will appear to alert the user to check the contact number input or ask them to register the pet owner first before registering the pet details.**

**Procedure code:**

CREATE OR REPLACE Procedure Prc\_register\_pet(in\_owner\_Contact IN VARCHAR2, in\_pet\_Name IN VARCHAR2, in\_pet\_dob IN Date, in\_pet\_type IN CHAR) AS

No\_owner\_found EXCEPTION;

PRAGMA exception\_init(No\_owner\_found,-20201);

v\_petOwner\_id PetOwner.owner\_id%TYPE;

v\_owner\_name PetOwner.owner\_name%TYPE;

v\_pet\_id Pet.pet\_id%TYPE;

v\_sequence NUMBER;

BEGIN

SELECT owner\_id, owner\_name INTO v\_petOwner\_id, v\_owner\_name

FROM PetOwner

WHERE owner\_contact = in\_owner\_Contact;

IF SQL%FOUND THEN

SELECT pet\_seq.nextVal INTO v\_sequence FROM dual;

v\_pet\_id := TO\_CHAR('P'||v\_sequence);

Insert into Pet values(v\_pet\_id, v\_petOwner\_id, in\_pet\_Name, in\_pet\_dob, in\_pet\_type);

dbms\_output.put\_line('The pet has been inserted.');

dbms\_output.put\_line(chr(10));

dbms\_output.put\_line('PET REGISTER DETAIL');

dbms\_output.put\_line((LPAD('=',20,'=')));

dbms\_output.put\_line('Owner ID :'||v\_petOwner\_id);

dbms\_output.put\_line('Owner Name :'||v\_owner\_name);

dbms\_output.put\_line('Pet ID :'||v\_pet\_id);

dbms\_output.put\_line('Pet Name :'||in\_pet\_Name);

dbms\_output.put\_line('Pet Dob :'||in\_pet\_dob);

dbms\_output.put\_line('Pet Type :'||in\_pet\_type);

END IF;

EXCEPTION

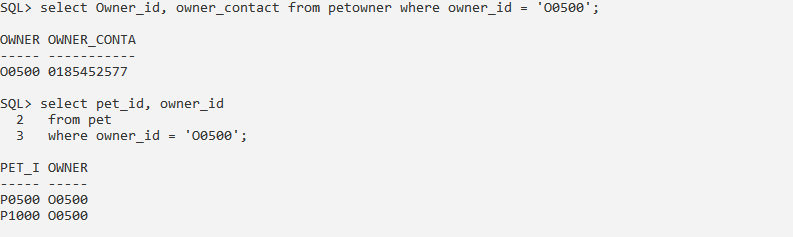
WHEN NO\_DATA\_FOUND THEN

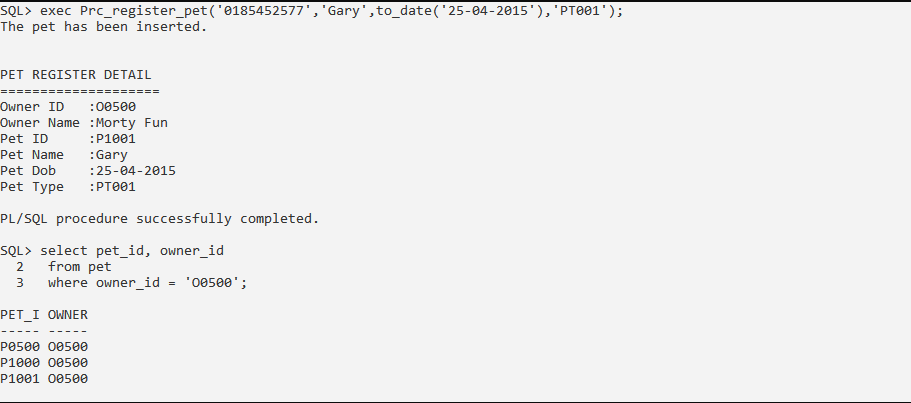
RAISE\_application\_error(-20201,'The owner is not exist! Check Phone No or create new owner.');

END;

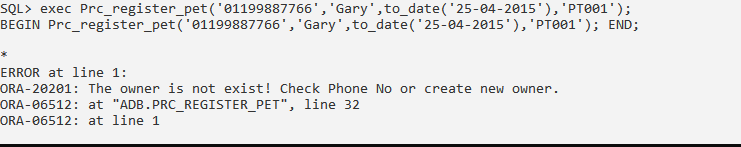
/

**Sample Output:**

****

****

**Exception**

****

**4.3.5 Procedure 2: Display Owner Information**

**Purpose: The purpose of this procedure is to list the owner's pet information. With this procedure, the staff can know all the pet details of the owner. For instance, last visit date for the pet and total visit quantity.**

**Procedure code:**

CREATE OR REPLACE Procedure Prc\_show\_owner\_info(in\_owner\_Contact IN VARCHAR2) AS

CURSOR owner\_list IS

SELECT o.owner\_id, o.owner\_name, p.pet\_id, p.pet\_name,

t.type\_name AS pet\_Type, MAX(appointment\_datetime) AS last\_Vist,

COUNT(appointment\_datetime) AS Total\_Visit

FROM PetOwner o, Pet p, Appointment a, PetType t

WHERE o.owner\_contact = 0185452577 AND

a.pet\_id = p.pet\_id AND

t.type\_id = p.type\_id AND

o.owner\_id = p.owner\_id

Group by o.owner\_id, o.owner\_name, p.pet\_id, p.pet\_name, t.type\_name

Order by 3;

owner\_r owner\_list%ROWTYPE;

v\_count NUMBER;

BEGIN

v\_count := 0;

OPEN owner\_list;

LOOP

FETCH owner\_list INTO owner\_r;

EXIT WHEN owner\_list%NOTFOUND;

IF (v\_count = 0) THEN

dbms\_output.put\_line(' OWNER DETAIL');

dbms\_output.put\_line((LPAD('=',22,'=')));

dbms\_output.put\_line('Owner ID :'||owner\_r.owner\_id);

dbms\_output.put\_line('Owner Name :'||owner\_r.owner\_name);

dbms\_output.put\_line(chr(10));

dbms\_output.put\_line(' PET DETAIL');

dbms\_output.put\_line((LPAD('=',22,'=')));

END IF;

dbms\_output.put\_line('Pet ID :'||owner\_r.pet\_id);

dbms\_output.put\_line('Pet Name :'||owner\_r.pet\_name);

dbms\_output.put\_line('Pet Type :'||owner\_r.pet\_Type);

dbms\_output.put\_line('Last Visit On :'||owner\_r.last\_Vist);

dbms\_output.put\_line('Total Visit :'||owner\_r.Total\_Visit);

dbms\_output.put\_line(chr(10));

v\_count := v\_count + 1;

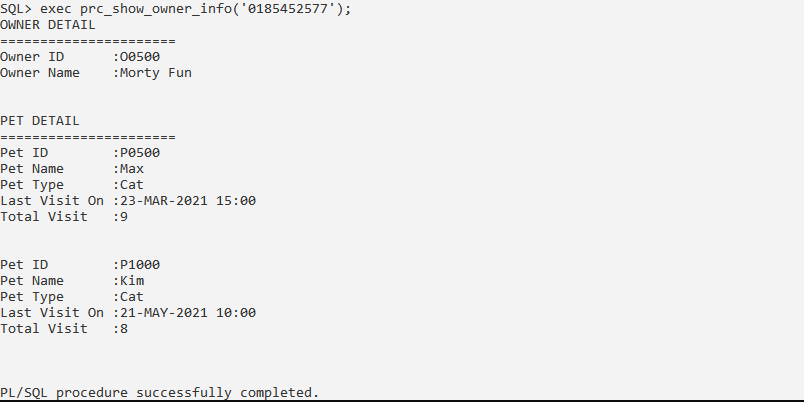
END LOOP;

CLOSE owner\_list;

END;

/

**Sample Output:**

****

**4.3.6 Trigger 1: Check Owner Age**

**Purpose: The purpose of this trigger is to check the owner's age. If the owner is less than 18, this trigger will occur and generate the error message and disallow the data insert into the database.**

**Trigger code:**

CREATE OR REPLACE TRIGGER trgOwnerAge

BEFORE INSERT OR UPDATE ON PetOwner

FOR EACH ROW

BEGIN

IF((ROUND((SYSDATE-:new.owner\_dob )/365)) < 18) THEN

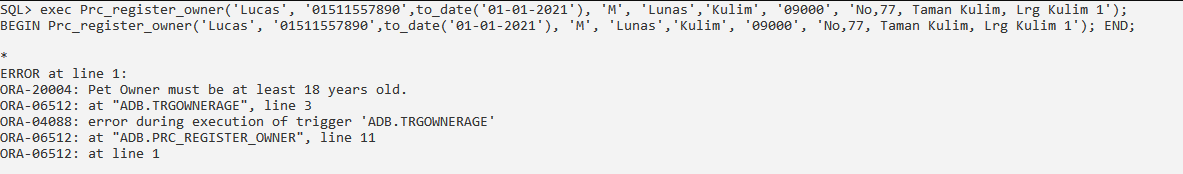
RAISE\_APPLICATION\_ERROR(-20004, 'Pet Owner must be at least 18 years old.' );

END IF;

END;

/

**Sample Output:**



**4.3.7 Trigger 2: Check Appointment Date Time**

**Purpose: The purpose of this trigger is to check the appointment date time. If the selected date and time for the selected vet has been appointed, this trigger will not allow the appointment to be made and prompt the suggested time to the user.**

**Trigger code:**

ALTER SESSION SET NLS\_DATE\_FORMAT = 'DD-MM-YYYY HH24:MI:SS';

CREATE OR REPLACE TRIGGER TRG\_CHK\_APPOINTMENT\_DATE

BEFORE INSERT ON Appointment

FOR EACH ROW

DECLARE

Date\_Time\_Booked EXCEPTION;

PRAGMA exception\_init(Date\_Time\_Booked, -20200 );

v\_tempDate DATE;

v\_startDate DATE;

v\_BOOKED\_APPOINTMENT DATE;

v\_appointment\_dateTime DATE;

v\_count NUMBER;

v\_check NUMBER;

v\_cursorLength NUMBER;

CURSOR c1 IS

SELECT appointment\_dateTime

FROM Appointment

WHERE EXTRACT(YEAR FROM(appointment\_dateTime)) = EXTRACT(YEAR FROM(:new.appointment\_dateTime)) AND

EXTRACT(Month FROM(appointment\_dateTime)) = EXTRACT(Month FROM(:new.appointment\_dateTime)) AND

EXTRACT(DAY FROM(appointment\_dateTime)) = EXTRACT(DAY FROM(:new.appointment\_dateTime)) AND

vet\_id = :new.vet\_id;

CURSOR c2 IS

SELECT appointment\_dateTime

FROM Appointment

WHERE EXTRACT(YEAR FROM(appointment\_dateTime)) = EXTRACT(YEAR FROM(:new.appointment\_dateTime)) AND

EXTRACT(Month FROM(appointment\_dateTime)) = EXTRACT(Month FROM(:new.appointment\_dateTime)) AND

EXTRACT(DAY FROM(appointment\_dateTime)) = EXTRACT(DAY FROM(:new.appointment\_dateTime)) AND

vet\_id = :new.vet\_id;

BEGIN

v\_startDate := TRUNC(:new.appointment\_dateTime);

v\_count := 0;

v\_cursorLength := 0;

SELECT a.appointment\_dateTime into V\_appointment\_dateTime

FROM Appointment a

WHERE a.appointment\_dateTime = :new.appointment\_dateTime AND

a.vet\_id = :new.vet\_id;

IF SQL%FOUND THEN

dbms\_output.put\_line('The time has been book.');

OPEN c1;

LOOP

FETCH c1 INTO v\_tempDate;

EXIT WHEN c1%NOTFOUND;

v\_cursorLength := v\_cursorLength+1;

END LOOP;

IF v\_cursorLength > 7 THEN

dbms\_output.put\_line('The vet is fulled on this date. Please choose other vet or change date.');

RAISE\_application\_error(-20200,'The Date Time has been booked');

END IF;

v\_startDate := v\_startDate + 10/24;

dbms\_output.put\_line('Suggested Date Time');

dbms\_output.put\_line('===================');

WHILE v\_count <= 7

LOOP

v\_check := 0;

OPEN c2;

LOOP

FETCH c2 INTO v\_booked\_appointment;

EXIT WHEN c2%NOTFOUND;

IF (v\_startDate != v\_booked\_appointment AND v\_check <= v\_cursorLength) THEN

v\_check := v\_check + 1;

END IF;

END LOOP;

CLOSE c2;

IF v\_check = v\_cursorLength THEN

dbms\_output.put\_line(v\_startDate);

END IF;

v\_count := v\_count + 1;

v\_startDate := v\_startDate + 1/24;

END LOOP;

dbms\_output.put\_line('===================');

dbms\_output.put\_line('Available time slot on '|| v\_startDate ||' for the selected vet is shown above.' );

RAISE\_application\_error(-20200,'The Date Time has been booked');

END IF;

EXCEPTION

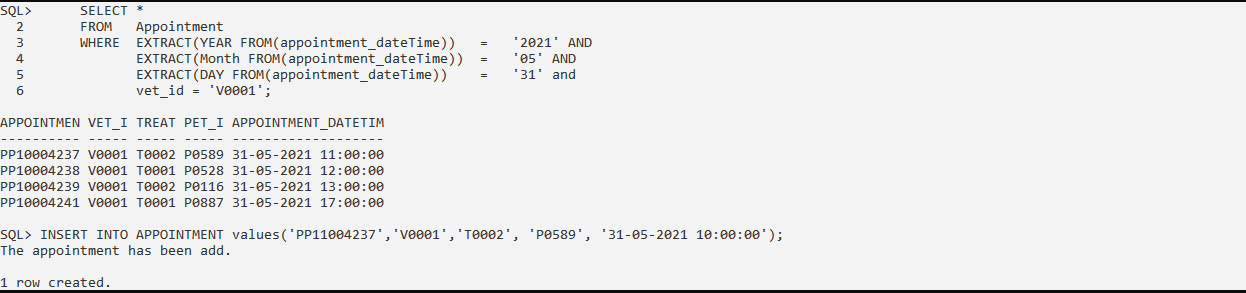
WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line('The appointment has been add.');

END;

/

**Sample Output:**



**4.3.8 Report 1: Summary Report of Treatment Revenue in Selected Year**

**Purpose: The purpose of this summary report is to show the treatment revenue for the selected year. This report can help the management to understand the total treatment quantity and revenue for each branch in the selected year. This report can help the management to know which treatment is most famous or profitable in each branch. This report can help the management to gain the insight of the treatment for each branch.**

**Report code:**

set linesize 175;

set pagesize 200;

e\_invalid\_year EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_year, -20166);

CURSOR pet\_detail\_list IS

CREATE OR REPLACE PROCEDURE rpt\_treatment\_revenue(IN\_YEAR IN NUMBER) AS

CURSOR treatment\_list IS

SELECT b.branch\_id, t.treatment\_id, t.treatment\_type as treatment\_name,

t.treatment\_price as Price,

COUNT(t.treatment\_price) as qty,

SUM(t.treatment\_price) as Total\_Treatment\_Earn

FROM Treatment t, Appointment a, Branch b, Transaction v

WHERE b.branch\_id = v.branch\_id AND

v.appointment\_id = a.appointment\_id AND

a.treatment\_id = t.treatment\_id AND

EXTRACT(YEAR FROM(v.transaction\_dateTime)) = IN\_YEAR

GROUP BY b.branch\_id, t.treatment\_id, t.treatment\_type,

t.treatment\_price

ORDER BY 1;

treatment\_list\_r treatment\_list%ROWTYPE;

v\_count NUMBER;

v\_revenue NUMBER;

v\_total\_revenue NUMBER;

v\_branch\_id VARCHAR2(5);

BEGIN

IF IN\_YEAR > Extract(Year From(sysdate)+1) or IN\_YEAR < 2019 THEN

RAISE\_application\_error(-20166,'Invalid year');

END IF;

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 50, ' ') || 'Summary Treatment Revenue Report in '|| IN\_YEAR);

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 50, ' ') || RPAD('\_', 40,'\_'));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 72, ' ') || 'Report generated on : ' ||

TO\_CHAR(CURRENT\_DATE, 'DD-MM-YYYY HH:MI:SS ') ||

'by ' || USER);

DBMS\_OUTPUT.PUT\_LINE(chr(10));

v\_count := 0;

v\_revenue :=0;

v\_total\_revenue :=0;

v\_branch\_id := ' ';

OPEN treatment\_list;

LOOP

FETCH treatment\_list INTO treatment\_list\_r;

EXIT WHEN treatment\_list%NOTFOUND;

IF v\_count > 0 AND v\_branch\_id != treatment\_list\_r.branch\_id THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 120, '-'));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('--------------------', 75, ' ') ||'--------------------');

DBMS\_OUTPUT.PUT\_LINE(RPAD('Total Earn :', 83, ' ') ||'RM ' || v\_revenue);

DBMS\_OUTPUT.PUT\_LINE(RPAD('--------------------', 75, ' ') ||'--------------------');

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

v\_total\_revenue := v\_total\_revenue + v\_revenue;

v\_revenue :=0;

END IF;

IF v\_count = 0 OR v\_branch\_id != treatment\_list\_r.branch\_id THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 30, '-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Branch ID', 15, ' ') || ': ' ||

RPAD(UPPER(treatment\_list\_r.branch\_id), 10, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 30, '-'));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 120, '='));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Treatment ID', 20, ' ') ||

RPAD('Treatment Name', 30, ' ') ||

RPAD('Price',18, ' ') ||

RPAD('Quantity',15, ' ') ||

RPAD('Revenue',20, ' ')

);

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 120, '='));

END IF;

DBMS\_OUTPUT.PUT\_LINE(RPAD(treatment\_list\_r.treatment\_id, 20, ' ')||

RPAD(treatment\_list\_r.treatment\_name, 30, ' ')||

'RM ' ||RPAD(treatment\_list\_r.Price, 15, ' ')||

RPAD(treatment\_list\_r.qty, 15, ' ')||

'RM ' ||RPAD(treatment\_list\_r.Total\_Treatment\_Earn, 20, ' ')

);

v\_count := v\_count + 1;

v\_revenue := v\_revenue + treatment\_list\_r.Total\_Treatment\_Earn;

v\_branch\_id := treatment\_list\_r.branch\_id;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 120, '-'));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('--------------------', 77, ' ') ||'--------------------');

DBMS\_OUTPUT.PUT\_LINE(RPAD('Total Earn :', 83, ' ') ||'RM ' || v\_revenue);

DBMS\_OUTPUT.PUT\_LINE(RPAD('--------------------', 77, ' ') ||'--------------------');

DBMS\_OUTPUT.PUT\_LINE(chr(10));

v\_total\_revenue := v\_total\_revenue + v\_revenue;

DBMS\_OUTPUT.PUT\_LINE(RPAD('=========================', 77, ' ') ||'====================');

DBMS\_OUTPUT.PUT\_LINE(RPAD('Total Treatment Revenue: ', 83, ' ') || 'RM ' || v\_total\_revenue);

DBMS\_OUTPUT.PUT\_LINE(RPAD('=========================', 77, ' ') ||'====================');

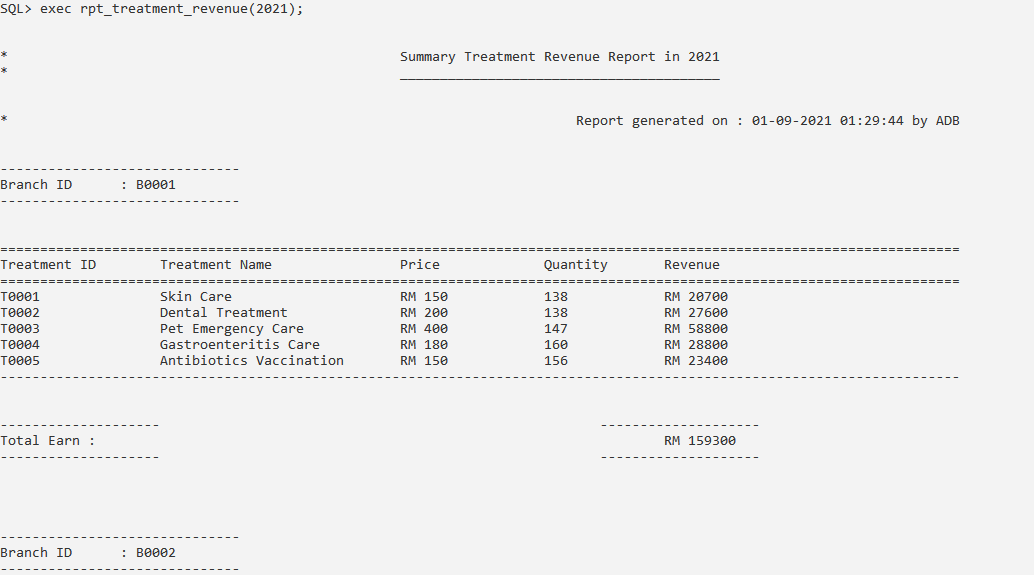
DBMS\_OUTPUT.PUT\_LINE(chr(10));

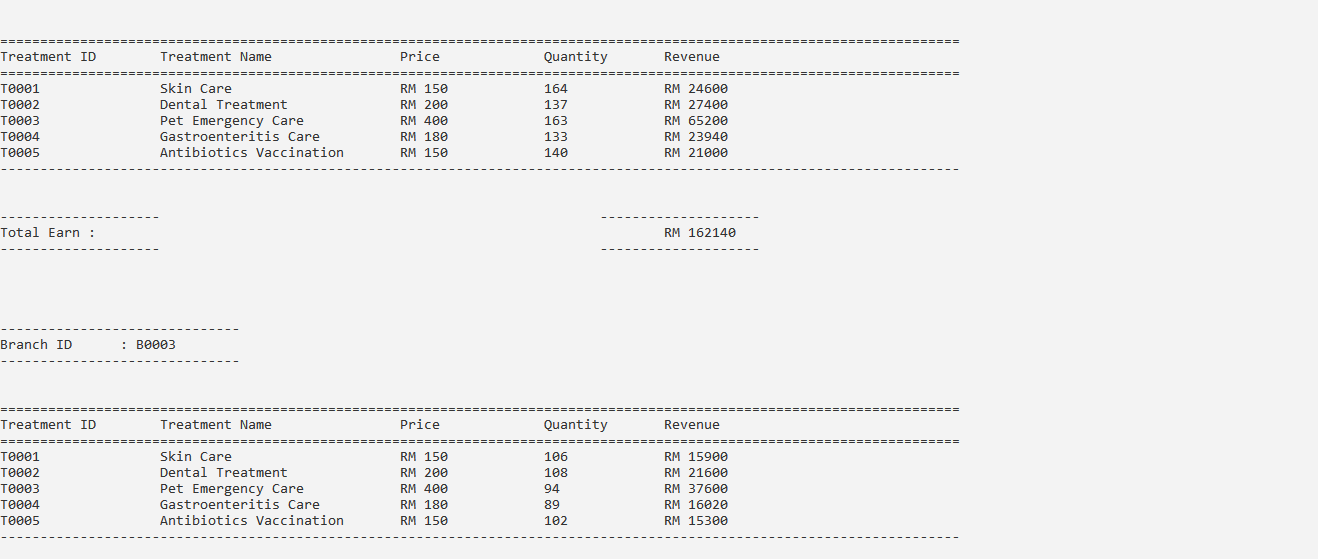
DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 92, ' ')||'End of report');

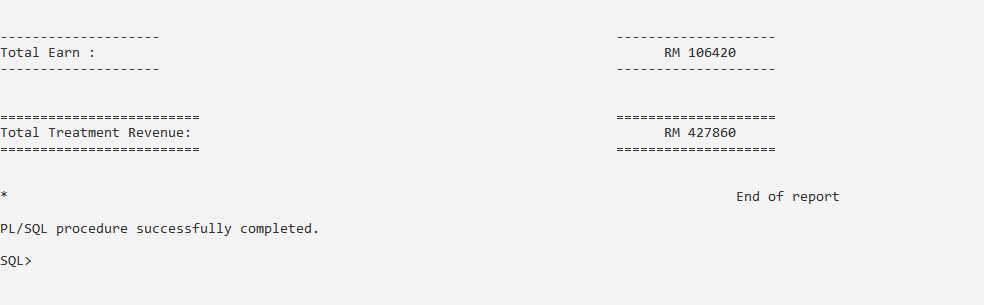
END;

/

**Sample Output:**







**4.3.9 Report 2: Detail Report of Medic Performance**

**Purpose: The purpose of this detailed report is to show all medic performance from the start of the business until today. It will show the total purchase, total sales, stock quantity, gross profit and gross profit margin for each medicine. This report can help the management to know the performance of their medicine.**

**The calculation is show below:**

**Stock Quantity = Purchase quantity - sold quantity**

**Revenue = Sales price \* sold quantity**

**Cost of good sold(COSG) = Purchase price \* sold quantity**

**Gross profit = Revenue - COSG**

**Gross profit margin = Gross profit / Revenue \* 100**

**Report code:**

ALTER SESSION SET NLS\_DATE\_FORMAT = 'DD-MM-YYYY';

set linesize 170;

set pagesize 200;

CREATE OR REPLACE PROCEDURE rpt\_medic\_performance AS

CURSOR medic\_list IS

SELECT s.supplier\_id, s.supplier\_name, s.supplier\_contact, m.medic\_id,

m.medic\_name, i.purchase\_price, SUM(i.purchase\_qty) as Purchase\_qty,

m.medic\_price as Sales\_price, m.medic\_qty as stock,

(SUM(i.purchase\_qty)- m.medic\_qty) as Sold\_qty

FROM Supplier s, PurchaseTransaction p, PurchaseItem i,

MedicalSupply m

WHERE s.supplier\_id = p.supplier\_id AND

p.purchase\_id = i.purchase\_id AND

i.medic\_id = m.medic\_id

GROUP BY s.supplier\_id, s.supplier\_name, s.supplier\_contact,

m.medic\_id, m.medic\_name, i.purchase\_price,

m.medic\_price, m.medic\_qty

ORDER BY 1;

medic\_list\_r medic\_list%ROWTYPE;

v\_supplier\_id VARCHAR(5);

v\_count NUMBER;

v\_count2 NUMBER;

grossProfit NUMBER(11,2);

grossProfitMargin NUMBER(11,2);

revenue NUMBER(15,2);

COGS NUMBER(15,2);

totalGrossProfit NUMBER(15,2);

BEGIN

v\_count := 0;

v\_count2 := 0;

grossProfit := 0;

grossProfitMargin := 0;

revenue := 0;

COGS := 0;

v\_supplier\_id := ' ';

totalGrossProfit := 0;

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 60, ' ') || 'Detail Medic Performance Report');

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 60, ' ') || 'Since 01-01-2019' ||' to '|| sysdate);

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 60, ' ') || RPAD('\_', 31,'\_'));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 92, ' ') || 'Report generated on : ' ||

TO\_CHAR(CURRENT\_DATE, 'DD-MM-YYYY HH:MI:SS ') ||

'by ' || USER);

DBMS\_OUTPUT.PUT\_LINE(chr(10));

OPEN medic\_list;

LOOP

FETCH medic\_list INTO medic\_list\_r;

EXIT WHEN medic\_list%NOTFOUND;

IF v\_count > 0 AND v\_supplier\_id != medic\_list\_r.supplier\_id THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 165, '-'));

DBMS\_OUTPUT.PUT\_LINE('.'||LPAD(v\_count2||' records found for supplier ' || v\_supplier\_id|| '.', 155, ' '));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

totalgrossprofit := 0;

v\_count2 := 0;

END IF;

IF v\_count = 0 OR v\_supplier\_id != medic\_list\_r.supplier\_id THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 16, '-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Supplier Detail', 15, ' ') || ': ' ||

RPAD(UPPER(medic\_list\_r.supplier\_id), 8, ' ') ||

RPAD(UPPER(medic\_list\_r.supplier\_name), 30, ' ') ||

RPAD(UPPER(medic\_list\_r.supplier\_contact), 15, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 16, '-'));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 165, '='));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Medical', 37, ' ') ||

RPAD('Purchase Cost',15, ' ') ||

RPAD('Sales Price',15, ' ') ||

RPAD('Total Purchase',17, ' ') ||

RPAD('Sold Qty',10, ' ') ||

RPAD('Stock',8, ' ') ||

RPAD('Revenue',15, ' ') ||

RPAD('COGS',15, ' ') ||

RPAD('Gross Profit',15, ' ') ||

RPAD('GP Margin %',10, ' ')

);

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 165, '='));

END IF;

revenue := medic\_list\_r.Sales\_price\* medic\_list\_r.Sold\_qty;

COGS := medic\_list\_r.purchase\_price \* medic\_list\_r.Sold\_qty;

grossProfit := revenue - COGS;

grossProfitMargin := grossProfit / revenue \* 100;

DBMS\_OUTPUT.PUT\_LINE(RPAD(medic\_list\_r.medic\_id, 7, ' ') ||

RPAD(medic\_list\_r.medic\_name, 30, ' ') ||

RPAD('RM'||(TRIM(TO\_CHAR(medic\_list\_r.purchase\_price, '9999D99'))), 15, ' ') ||

RPAD('RM'||(TRIM(TO\_CHAR(medic\_list\_r.Sales\_price, '9999D99'))), 15, ' ') ||

RPAD(medic\_list\_r.purchase\_qty,17, ' ') ||

RPAD(medic\_list\_r.Sold\_qty,10, ' ') ||

RPAD(medic\_list\_r.stock,8, ' ') ||

RPAD('RM'||(TRIM(TO\_CHAR(revenue, '999999999D99'))), 15, ' ') ||

RPAD('RM'||(TRIM(TO\_CHAR(COGS, '999999999D99'))), 15, ' ') ||

RPAD('RM'||(TRIM(TO\_CHAR(grossprofit, '999999999D99'))), 15, ' ') ||

RPAD(TRIM(TO\_CHAR(grossProfitMargin, '999D99')) || '%', 10, ' ')

);

v\_count := v\_count + 1;

v\_count2 := v\_count2 + 1;

revenue := 0;

COGS := 0;

v\_supplier\_id := medic\_list\_r.supplier\_id;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 165, '-'));

DBMS\_OUTPUT.PUT\_LINE('.'||LPAD(v\_count2||' records found for supplier ' || v\_supplier\_id|| '.', 155, ' '));

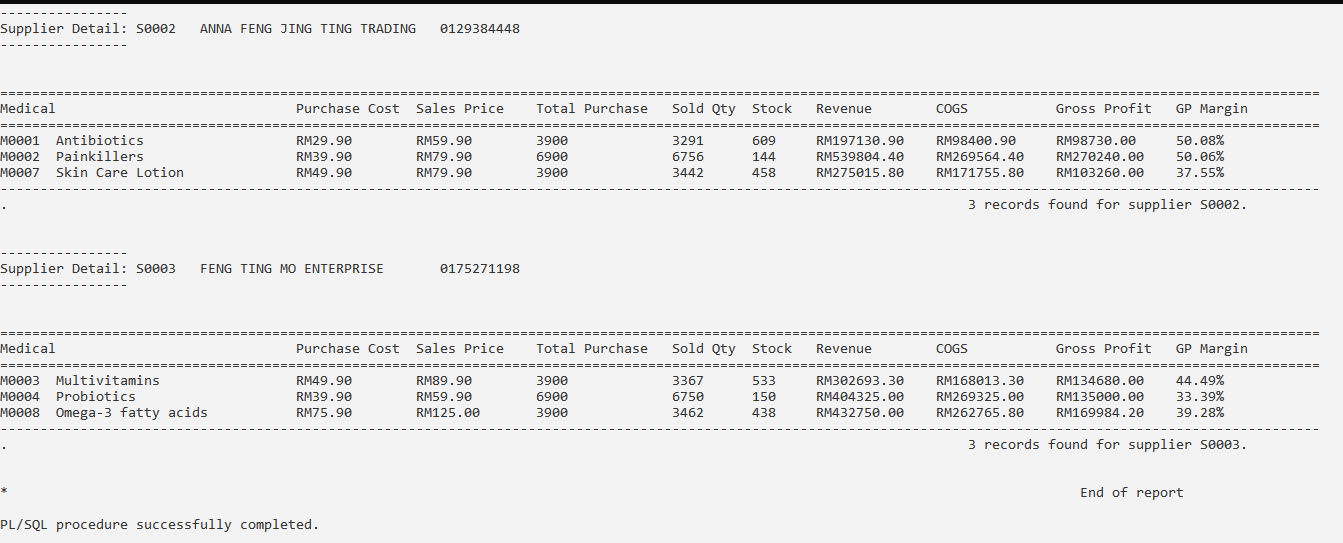
DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 135, ' ')||'End of report');

END;

/

**Sample Output:**

****

**4.3.10 Report 3: On-demand Report of the Pet Treatment Detail**

**Purpose: The purpose of this on-demand report is to let the veterinarian view the pet treatment history. This report will show all treatment, treatment date and handle veterinarian for the selected pet. It can help the veterinarian to know the condition of the pet and provide the most suitable treatment or medicine for it.**

**Report code:**

ALTER SESSION SET NLS\_DATE\_FORMAT = 'DD-MM-YYYY HH24:MI:SS';

set linesize 125;

CREATE OR REPLACE PROCEDURE rpt\_pet\_treatment\_detail(IN\_PETID IN CHAR) AS

NO\_PET\_FOUND EXCEPTION;

PRAGMA EXCEPTION\_INIT(NO\_PET\_FOUND, -20202);

CURSOR pet\_detail\_list IS

SELECT o.owner\_id, o.owner\_name, p.pet\_id,

p.pet\_name,t.type\_name AS pet\_Type,

a.appointment\_id,

a.appointment\_datetime, m.treatment\_id,

m.treatment\_type AS treatment\_name, v.vet\_id, v.vet\_name

FROM PetOwner o, Pet p, PetType t,

Veterinarian v,Appointment a,

Treatment m

WHERE o.owner\_id = p.owner\_id AND

p.type\_id = t.type\_id AND

p.pet\_id = a.pet\_id AND

p.pet\_id = UPPER(IN\_PETID) AND

a.vet\_id = v.vet\_id AND

m.treatment\_id = a.treatment\_id

ORDER BY 7;

pet\_detail\_r pet\_detail\_list%ROWTYPE;

v\_count NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 50, ' ') || 'Pet Treatment Detail Report');

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 50, ' ') || RPAD('\_', 30,'\_'));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 72, ' ') || 'Report generated on : ' ||

TO\_CHAR(CURRENT\_DATE, 'DD-MM-YYYY HH:MI:SS ') ||

'by ' || USER);

DBMS\_OUTPUT.PUT\_LINE(chr(10));

v\_count := 0;

OPEN pet\_detail\_list;

LOOP

FETCH pet\_detail\_list INTO pet\_detail\_r;

EXIT WHEN pet\_detail\_list%NOTFOUND;

IF (v\_count = 0) THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 55, '-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Owner Detail', 15, ' ') || ': ' ||

RPAD(UPPER(pet\_detail\_r.owner\_id), 10, ' ') ||

RPAD(UPPER(pet\_detail\_r.owner\_name), 40, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 55, '-'));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Pet Detail', 15, ' ' ) || ': ' ||

RPAD(UPPER(pet\_detail\_r.pet\_id), 10, ' ')||

RPAD(UPPER(pet\_detail\_r.pet\_name), 40, ' ')||

RPAD('Type', 10, ' ' ) || ': ' ||

RPAD(UPPER(pet\_detail\_r.pet\_Type), 20, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 120, '='));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Appointment ID', 20, ' ') ||

RPAD('Appointment Date Time', 25, ' ') ||

RPAD('Treatment ID',15, ' ') ||

RPAD('Treatment Name',30, ' ') ||

RPAD('Handle By:',40, ' ')

);

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 120, '='));

END IF;

DBMS\_OUTPUT.PUT\_LINE(RPAD(pet\_detail\_r.appointment\_id, 20, ' ')||

RPAD(pet\_detail\_r.appointment\_datetime, 25, ' ')||

RPAD(pet\_detail\_r.treatment\_id, 15, ' ')||

RPAD(pet\_detail\_r.treatment\_name, 30, ' ')||

RPAD(pet\_detail\_r.vet\_id, 7, ' ')||

RPAD(pet\_detail\_r.vet\_name, 30, ' ')

);

v\_count := v\_count + 1;

END LOOP;

IF v\_count = 0 THEN

RAISE\_application\_error(-20202,'No pet found');

END IF;

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 120, '-'));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('====================', 85, ' ') ||'====================');

DBMS\_OUTPUT.PUT\_LINE(RPAD('Total Appointment :', 95, ' ') || v\_count);

DBMS\_OUTPUT.PUT\_LINE(RPAD('====================', 85, ' ') ||'====================');

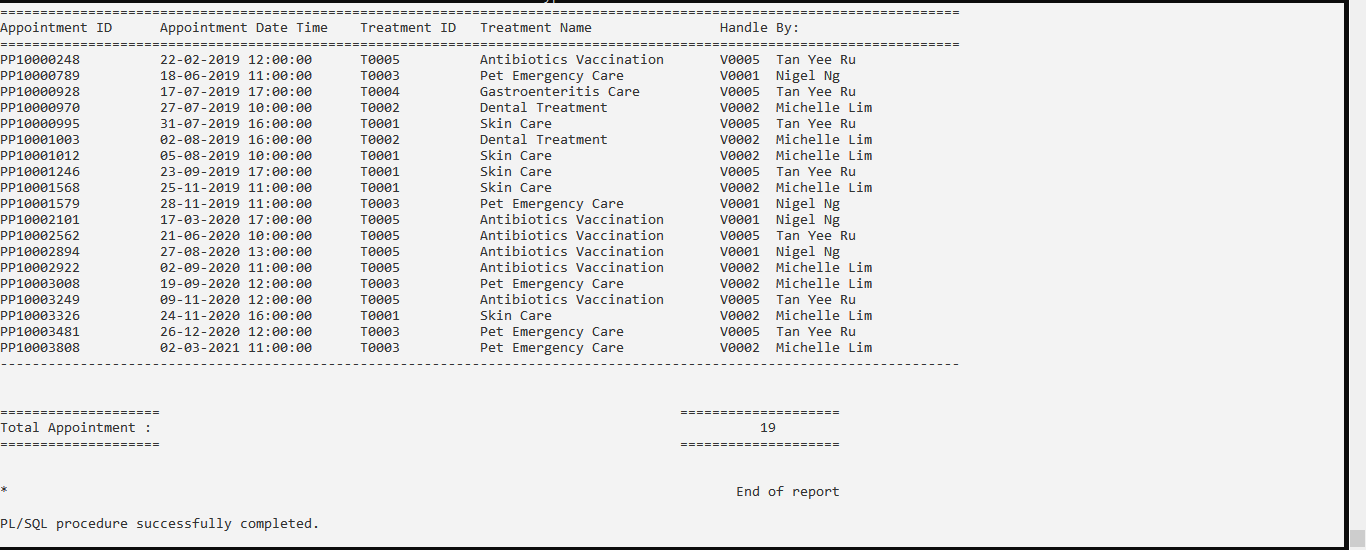
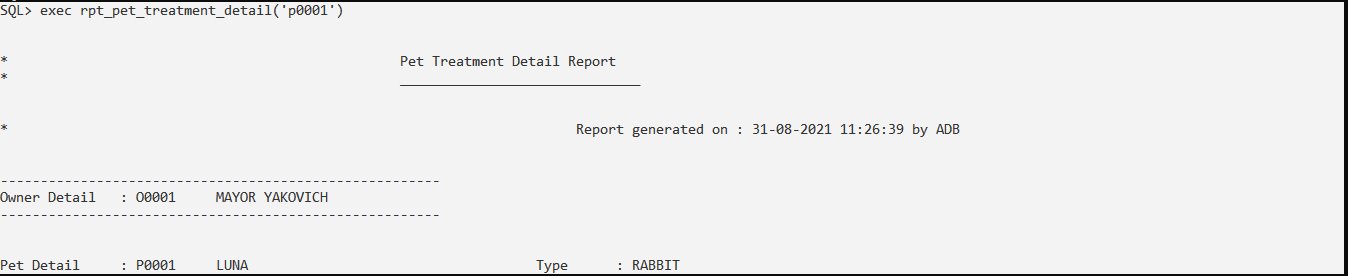
DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*', 92, ' ')||'End of report');

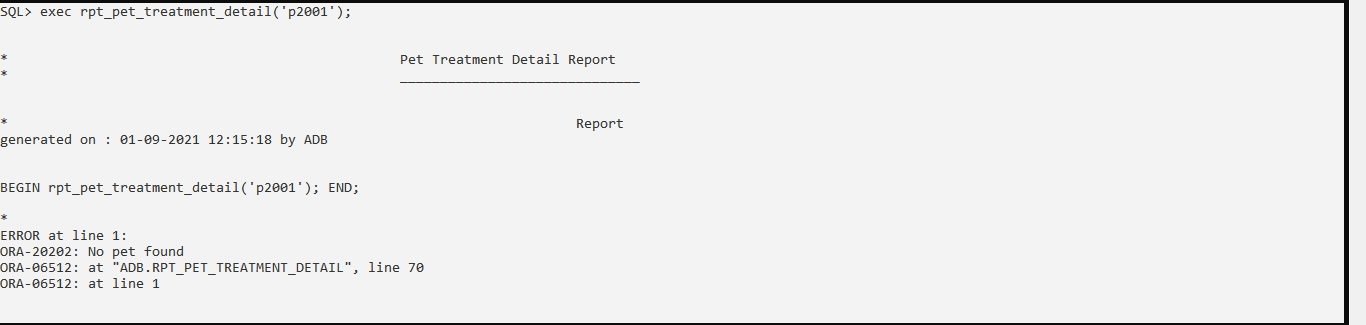
END;

/

**Sample Output:**

****

**Exception:**

****

### 4.4 (Nigel Lee Jian Hsee)

**4.4.1 Query 1: Top Three Pets Owner’s Postcode for each Branch based on total**

**transaction amount (Strategic)**

**Purpose: The purpose of this query is to find out where the clinic’s pet’s owners are mostly from each branch. This is able to help top management level staff or the founder of this clinic to make the decision of opening a new branch or moving their current branch in the future which is where most of their owners come from. The query ranks the top three owners’s postcodes based on the transaction amount from each pet’s owner postcode.**

SQL statement:

set pagesize 200

set linesize 80

clear break

clear compute

break on state on branch\_id skip 1

COMPUTE SUM LABEL TOTAL OF TOTALSALES TOTAL\_ORDER on branch\_id

TTITLE ON

TTITLE CENTER 'Top 3 Customer Postcode Revenue for each Clinic Branch' SKIP 1-

CENTER ========================================================= SKIP 2

column branch\_id FORMAT a10

column branch\_id HEADING 'Branch ID'

column STATE FORMAT a15

column STATE HEADING 'Branch State'

column CITY FORMAT a15

column CITY HEADING 'Customer City'

column POSTCODE FORMAT a10

column TotalSales format 9999999.99

column TotalSales HEADING 'Total |Revenue |(RM) '

column TotalSales format 9999999.99

column TOTAL\_ORDER HEADING 'Total|Order'

column TOTAL\_ORDER format 9999

column RANK format 99

SELECT branch\_id,

state,

city,

postcode,

TotalSales, Total\_Order, Rank

FROM

( SELECT T.branch\_id,O.state,

O.city,

O.postcode,

sum(Total\_Amount) as TotalSales,

count(distinct transaction\_id) as Total\_Order,

ROW\_NUMBER() OVER (PARTITION BY O.state

ORDER BY sum(Total\_Amount)

DESC)AS Rank

FROM Transaction T, PetOwner O

Where T.owner\_id = O.owner\_id

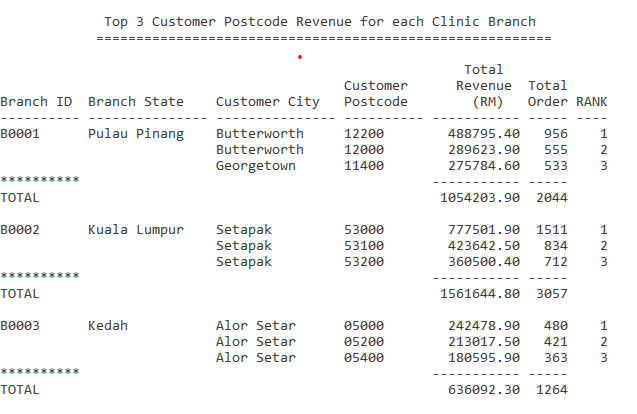
Group by T.branch\_id,O.state,O.city,O.postcode

)

WHERE Rank <= 3

ORDER BY branch\_id,TotalSales DESC;

**Sample Output:**



**4.4.2 Query 2: Rank top treatment, to medicine sales and total revenue of a**

**branch ()**

**Purpose: The purpose of this query is to rank the top treatment of a branch. This allows the branch clinic head or manager to know where the income from which type of treatments is. This query also shows that medicine revenue from each type of treatment. From this query, the branch head or manager can promote those treatments that have lower revenue.**

SQL statement:

set pagesize 200

set linesize 160

clear break

clear compute

break on state on branch\_id skip 1

COMPUTE SUM LABEL TOTAL OF Treatment\_Revenue Medic\_Revenue Total\_amount on branch\_id

TTITLE ON

TTITLE CENTER 'Year 2021 Treatment and Medicine Revenue of a Branch' SKIP 1-

CENTER ========================================================= SKIP 2

column TREATMENT\_TYPE HEADING 'Treatment Type'

column TREATMENT\_TYPE FORMAT A25

column TREATMENT\_REVENUE HEADING 'Treatment|Revenue|(RM)'

column TREATMENT\_REVENUE FORMAT 999999999.99

column MEDIC\_REVENUE HEADING 'Medic|Revenue|(RM)'

column MEDIC\_REVENUE FORMAT 999999999.99

column SOLD\_QUANTITY HEADING 'Sold|Medic|Quantity'

column SOLD\_QUANTITY FORMAT 99999

column Revenue\_Per\_Quantity HEADING 'Revenue|Per Medic|Quantity|(RM)'

column Revenue\_Per\_Quantity FORMAT 99999.99

column Total\_Amount HEADING 'Total|Revenue|(RM)'

column Total\_Amount FORMAT 9999999999.99

column treatment\_id HEADING 'Treatment|ID'

column Percent HEADING 'Percent|over|total|amount'

column Percent FORMAT 99.99

CREATE OR REPLACE VIEW FullRevenueGroupByTreatment2021 AS

select T.branch\_id,TT.treatment\_id, TT.treatment\_type, sum(T.total\_amount) as Total\_Amount

from transaction T, appointment A, treatment TT

where A.appointment\_id = T.appointment\_id AND A.treatment\_id = TT.treatment\_id AND EXTRACT(YEAR FROM T.transaction\_datetime) = '2021'

group by T.branch\_id,TT.treatment\_id, TT.treatment\_type

order by branch\_id;

CREATE OR REPLACE VIEW MedicRevenueGroupByTreatment2021 AS

select T.branch\_id,TT.treatment\_id, TT.treatment\_type, sum(TD.line\_total) AS Medic\_Revenue, sum(TD.line\_qty) AS Sold\_Quantity, (sum(TD.line\_total)/sum(TD.line\_qty)) AS Revenue\_Per\_Quantity

from transactiondetail TD, transaction T, appointment A, treatment TT

where TD.transaction\_id = T.transaction\_id AND A.appointment\_id = T.appointment\_id AND A.treatment\_id = TT.treatment\_id AND EXTRACT(YEAR FROM T.transaction\_datetime) = '2021'

group by T.branch\_id,TT.treatment\_id, TT.treatment\_type

order by branch\_id;

select F.branch\_id,F.treatment\_id,F.treatment\_type, (F.Total\_Amount - M.Medic\_Revenue) AS Treatment\_Revenue,

((F.Total\_Amount - M.Medic\_Revenue) / F.Total\_Amount \* 100) AS Percent,

ROW\_NUMBER() OVER (ORDER BY (F.Total\_Amount - M.Medic\_Revenue) DESC)AS Rank,

M.Medic\_Revenue,

(M.Medic\_Revenue / F.Total\_Amount \* 100) AS Percent,

ROW\_NUMBER() OVER (ORDER BY M.Medic\_Revenue DESC)AS Rank,

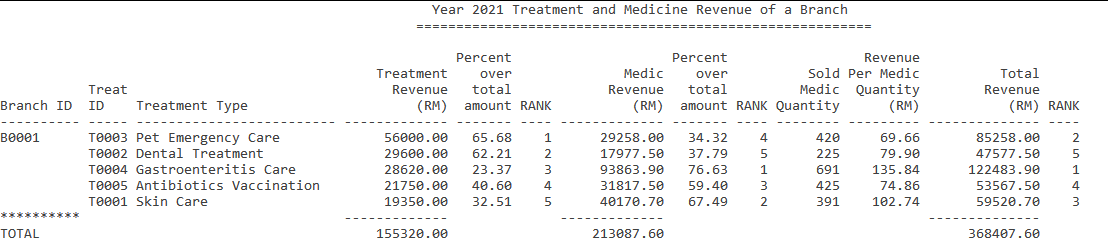
Sold\_Quantity,Revenue\_Per\_Quantity, F.Total\_amount,

ROW\_NUMBER() OVER (ORDER BY F.Total\_amount DESC)AS Rank

from FullRevenueGroupByTreatment2021 F, MedicRevenueGroupByTreatment2021 M

where F.branch\_id = '&branch\_id' AND F.branch\_id = M.branch\_id AND F.treatment\_id = M.treatment\_id;

**Sample Output:**



**4.4.3 Query 3: View Customer Transaction History by entering Customer ID (Operational)**

**Purpose: The purpose of this query is to check the previous transaction history by entering the owner ID. This clinic veterinarian checks the transaction history to view all treatments for all the owner’s pets or what medic they have bought previously for their pet. All medical history of the owner’s pets will also be shown.**

clear break

clear compute

transaction history of a customer

set pagesize 200

set linesize 200

alter session set nls\_date\_format = 'DD-MON-YYYY';

TTITLE ON

TTITLE CENTER 'Transaction History' SKIP 1-

CENTER ========================================================= SKIP 2

break on owner\_id on owner\_name on pet\_id on pet\_name on transaction\_datetime on transaction\_id on total\_amount on treatment\_price on treatment\_type skip 1

column OWNER\_ID HEADING 'Owner|ID'

column OWNER\_NAME HEADING 'Owner Name'

column OWNER\_NAME FORMAT A15

column PET\_ID HEADING 'Pet|ID'

column PET\_NAME HEADING 'Pet Name'

column PET\_NAME FORMAT A15

column TRANSACTION\_ID HEADING 'Transaction|ID'

column TRANSACTION\_ID FORMAT A11

column TRANSACTION\_DATETIME HEADING 'Transaction|Date'

column TREATMENT\_TYPE HEADING 'Treatment Type'

column TREATMENT\_TYPE FORMAT A25

column TREATMENT\_PRICE HEADING 'Treatment|Price|(RM)'

column TREATMENT\_PRICE FORMAT 999.99

column MEDIC\_ID HEADING 'Medic|ID'

column MEDIC\_NAME HEADING 'Medicine|Name'

column MEDIC\_PRICE HEADING 'Medicine|Unit|Price|(RM)'

column MEDIC\_PRICE FORMAT 999.99

column LINE\_QTY HEADING 'Quantity'

column LINE\_QTY FORMAT 99

column LINE\_TOTAL HEADING 'Line|Total|(RM)'

column LINE\_TOTAL FORMAT 9999.99

column Total\_Amount HEADING 'Transaction|Total|(RM)'

column Total\_Amount FORMAT 9999999999.99

select T.owner\_id,PO.owner\_name,P.pet\_id,P.pet\_name,T.transaction\_id,

T.transaction\_datetime,

TT.treatment\_type, TT.treatment\_price, M.medic\_id,M.medic\_name,

M.medic\_price, TD.line\_qty, TD.line\_total,T.total\_amount

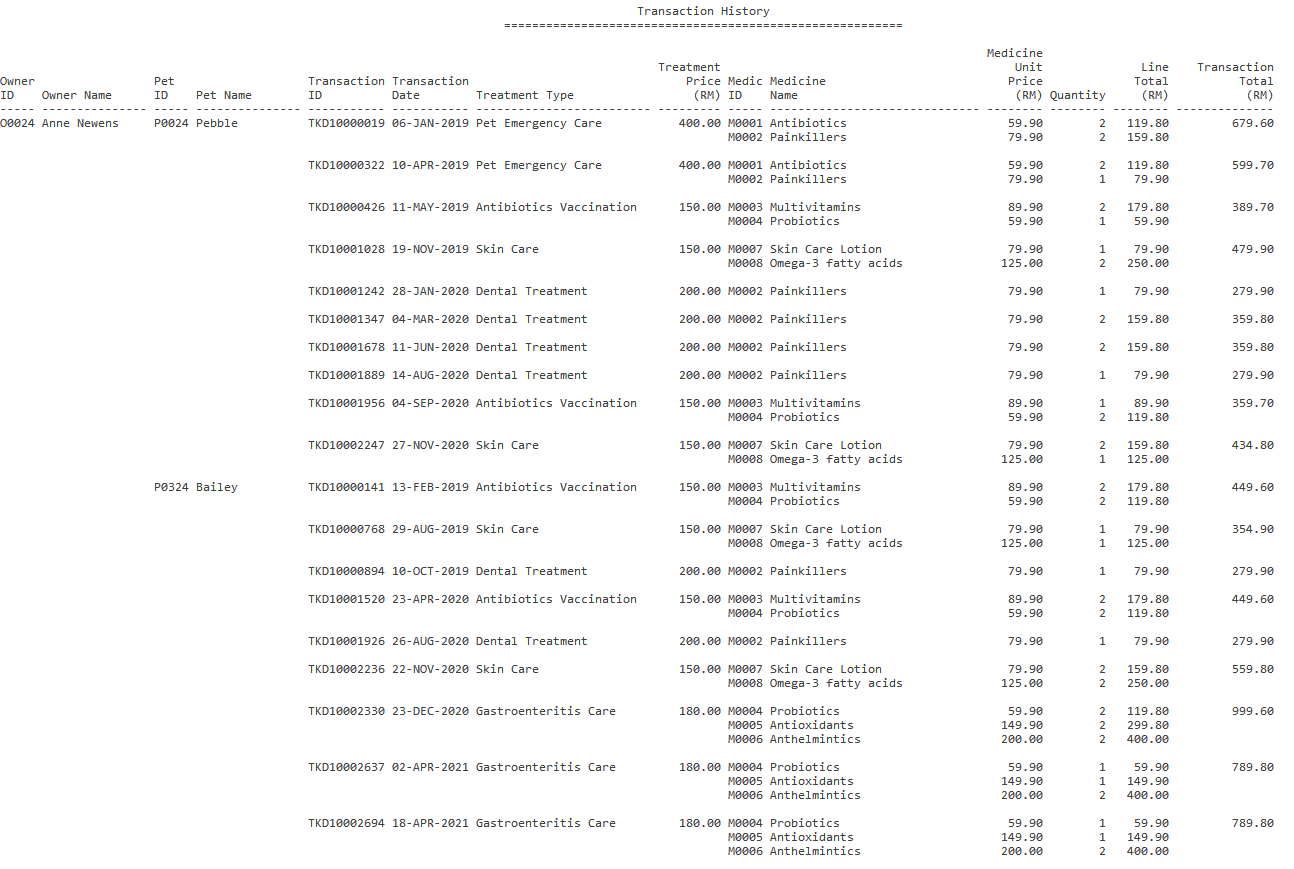
from medicalsupply M, transactiondetail TD, transaction T, appointment A, treatment TT,PetOwner PO, Pet P

where M.medic\_id = TD.medic\_id AND TD.transaction\_id = T.transaction\_id AND T.appointment\_id = A.appointment\_id

AND A.treatment\_id = TT.treatment\_id AND A.pet\_id = P.pet\_id

AND PO.owner\_id = T.Owner\_id AND PO.Owner\_id = '&Owner\_id'

order by pet\_id,transaction\_datetime, medic\_id;



#### 

#### 

**4.4.4 Procedure 1: Create Transaction**

**Purpose: The purpose of this procedure is to allow the staff or veterinarian to create a transaction after the appointment and treatment of the owner’s pet. The procedure will require the appointment id and will auto insert the owner id, branch id, update the total amount of the transition by adding the treatment price, and record the transaction date time. If any appointment id that entered is not found from the appointment table, an exception will be raised. Only existing appointments were able to make payment. The payment must be made after the appointment if not a trigger will be triggered and raise an exception. If the transaction of appointment has been made, an exception will be raised to notify the veterinarian that the transaction for this appointment has been created.**

**Procedure code:**

SET SERVEROUTPUT ON FORMAT WRAPPED

alter session set nls\_date\_format = 'DD-MON-YYYY HH24:MI';

CREATE OR REPLACE PROCEDURE payment\_module (IN\_appointment\_id in APPOINTMENT.appointment\_id%TYPE) IS

appointmentCount NUMBER :=0;

transactionCount NUMBER :=0;

v\_transaction\_id TRANSACTION.transaction\_id%TYPE;

v\_treatment\_id TREATMENT.treatment\_id%TYPE;

v\_treatment\_price TREATMENT.treatment\_price%TYPE;

v\_owner\_id TRANSACTION.owner\_id%TYPE;

v\_pet\_id PET.pet\_id%TYPE;

v\_branch\_id TRANSACTION.branch\_id%TYPE;

v\_datetime TRANSACTION.transaction\_datetime%TYPE;

v\_appointment\_time APPOINTMENT.appointment\_datetime%TYPE;

v\_totalamount TRANSACTION.total\_amount%TYPE;

v\_treatmenttype TREATMENT.treatment\_type%TYPE;

v\_pet\_name PET.pet\_name%TYPE;

v\_vet\_id APPOINTMENT.vet\_id%TYPE;

e\_invalid\_appointmentid EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_appointmentid, -20150);

e\_repeated\_transaction EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_repeated\_transaction, -20151);

BEGIN

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('4 Golden Duck Wellness Veterinary Clinic',55, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',70,'='));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Payment Module',40, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',70,'='));

select count(appointment\_id) INTO appointmentCount

from appointment

where appointment\_id = IN\_appointment\_id;

IF appointmentCount = 0 THEN

RAISE\_APPLICATION\_ERROR(-20150, 'Appointment ID not Found !!!', true);

END IF;

select count(transaction\_id) INTO transactionCount

from transaction

where appointment\_id = IN\_appointment\_id;

IF transactionCount > 0 THEN

RAISE\_APPLICATION\_ERROR(-20151, 'Repeated Transaction !!!', true);

END IF;

select treatment\_id, pet\_id, appointment\_datetime, vet\_id

INTO v\_treatment\_id, v\_pet\_id, v\_appointment\_time , v\_vet\_id

FROM appointment

WHERE appointment\_id = IN\_appointment\_id;

select treatment\_price, treatment\_type

INTO v\_treatment\_price,v\_treatmenttype

FROM treatment

WHERE treatment\_id = v\_treatment\_id;

select owner\_id, pet\_name

INTO v\_owner\_id,v\_pet\_name

FROM pet

WHERE pet\_id = v\_pet\_id;

select branch\_id

INTO v\_branch\_id

from veterinarian

WHERE vet\_id = v\_vet\_id;

select sysdate

into v\_datetime

from dual;

v\_totalamount := v\_treatment\_price;

v\_transaction\_id := TO\_CHAR('T'||IN\_appointment\_id);

DBMS\_OUTPUT.PUT\_LINE('Branch id : ' || v\_Branch\_id);

DBMS\_OUTPUT.PUT\_LINE('Transaction ID : '||v\_transaction\_id);

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE('Appointment id : ' || IN\_appointment\_id);

DBMS\_OUTPUT.PUT\_LINE('Treatment id : ' || v\_treatment\_id);

DBMS\_OUTPUT.PUT\_LINE('Treatment : ' || v\_treatmenttype);

DBMS\_OUTPUT.PUT\_LINE('Owner id : ' || v\_owner\_id);

DBMS\_OUTPUT.PUT\_LINE('Transaction Time : ' || TO\_CHAR(v\_datetime));

DBMS\_OUTPUT.PUT\_LINE('Pet ID : ' || v\_pet\_id);

DBMS\_OUTPUT.PUT\_LINE('Pet Name : ' || v\_pet\_name);

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',70,'-'));

DBMS\_OUTPUT.PUT\_LINE('Total Amount : RM' || to\_char (v\_totalamount, '9999.99'));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',70,'='));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Transaction Created',40, ' '));

Insert into transaction values(v\_transaction\_id,v\_owner\_id,IN\_appointment\_id,v\_Branch\_id,v\_totalamount,v\_datetime);

EXCEPTION

WHEN e\_invalid\_appointmentid THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',70,'-'));

DBMS\_OUTPUT.PUT\_LINE('Transaction Fail to Create !!! Entered Appointment does not exist !!!');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',70,'-'));

WHEN e\_repeated\_transaction THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',70,'-'));

DBMS\_OUTPUT.PUT\_LINE('The transaction for this appointment already exist !!!');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',70,'-'));

END;

/

exec payment\_module ('&appointment\_id')

#### Sample Output:

#### 

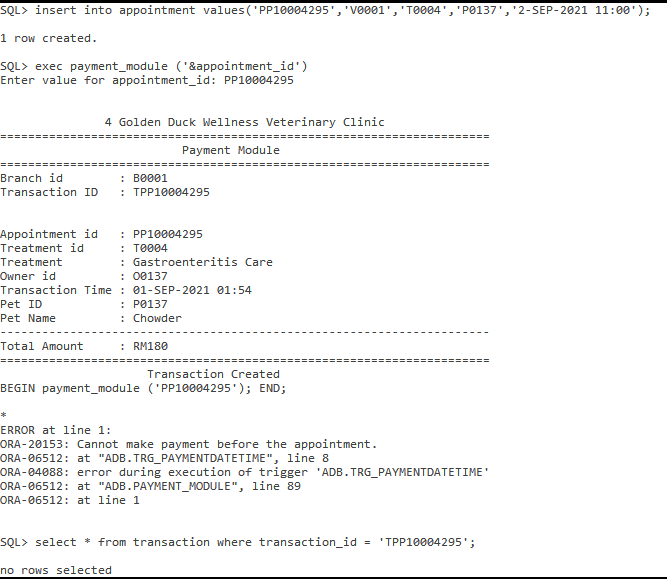
**Exception Output: Repeated appointment id**

#### 

**Exception Output: Invalid appointment ID**

#### 

**Exception Output: Triggered when make payment before appointment date time**

****

#### 

**4.4.5 Procedure 2: Add transaction detail**

**Purpose: The purpose of this procedure is to allow the staff or veterinarian to add transaction details such as medicine bought by the pet’s owner after creating transaction details. For example, adding medicine with id ‘M0001’ with quantity to a transaction. After adding transaction detail a trigger will be triggered to automatically update the total amount of the transaction. This procedure consists of validating whether the transaction id exists, the quantity must be more than zero, whether there is repeated medicine in the transaction.**

**Procedure code:**

SET SERVEROUTPUT ON FORMAT WRAPPED

alter session set nls\_date\_format = 'DD-MON-YYYY HH24:MI';

CREATE OR REPLACE PROCEDURE addTransactionDetail\_module (in\_transaction\_id in TRANSACTION.transaction\_id%TYPE, in\_medic\_id in TRANSACTIONDETAIL.transaction\_id%TYPE, in\_qty in TRANSACTIONDETAIL.line\_qty%TYPE) IS

transactionCount NUMBER :=0;

medicCount NUMBER :=0;

v\_treatmenttype TREATMENT.treatment\_type%TYPE;

v\_treatment\_price TREATMENT.treatment\_price%TYPE;

v\_medic\_price MEDICALSUPPLY.medic\_price%TYPE;

v\_medic\_price2 MEDICALSUPPLY.medic\_price%TYPE;

v\_medic\_name MEDICALSUPPLY.medic\_name%TYPE;

v\_medic\_name2 MEDICALSUPPLY.medic\_name%TYPE;

v\_line\_total TRANSACTIONDETAIL.line\_total%TYPE;

v\_transaction\_id TRANSACTIONDETAIL.transaction\_id%TYPE;

v\_totalamount TRANSACTION.total\_amount%TYPE;

v\_transactiondate TRANSACTION.transaction\_datetime%TYPE;

v\_daydifferent number;

e\_invalid\_transactionid EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_transactionid, -20154);

e\_repeated\_medicid EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_repeated\_medicid, -20155);

e\_zero\_qty EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_zero\_qty, -20156);

e\_dayexceed EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_dayexceed, -20157);

CURSOR detail\_cursor IS

SELECT \*

FROM transactiondetail

WHERE transaction\_id = in\_transaction\_id;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Transaction Detail',50, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

select count(transaction\_id)

INTO transactionCount

FROM transaction

WHERE transaction\_id = in\_transaction\_id;

IF transactionCount = 0 THEN

RAISE\_APPLICATION\_ERROR(-20154, 'Transaction ID not Found and Not Created yet !!!', true);

END IF;

SELECT SYSDATE - transaction\_datetime, transaction\_datetime INTO v\_daydifferent,v\_transactiondate

FROM DUAL, transaction

where transaction\_id = in\_transaction\_id ;

IF v\_daydifferent > 7 THEN

RAISE\_APPLICATION\_ERROR(-20157, 'Day Exceed', true);

END IF;

IF in\_qty <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20156, 'More than zero', true);

END IF;

select count(transaction\_id)

INTO transactionCount

FROM transaction

WHERE transaction\_id = in\_transaction\_id;

IF transactionCount = 0 THEN

RAISE\_APPLICATION\_ERROR(-20154, 'Transaction ID not Found and Not Created yet !!!', true);

END IF;

select count(medic\_id)

INTO medicCount

FROM transactionDetail

WHERE medic\_id = in\_medic\_id AND transaction\_id = in\_transaction\_id;

IF medicCount >0 THEN

RAISE\_APPLICATION\_ERROR(-20155, 'The Medic for this transaction already existed !!!', true);

END IF;

select medic\_price, medic\_name

INTO v\_medic\_price, v\_medic\_name

FROM medicalsupply

WHERE medic\_id = in\_medic\_id;

v\_line\_total := in\_qty \* v\_medic\_price;

select treatment\_type, treatment\_price into v\_treatmenttype, v\_treatment\_price

from transaction T, appointment A, Treatment R

where T.transaction\_id = in\_transaction\_id AND T.appointment\_id = A.appointment\_id AND A.treatment\_id = R.treatment\_id;

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('4 Golden Duck Wellness Veterinary Clinic',65, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Payment Module',50, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(TO\_CHAR(v\_transactiondate));

DBMS\_OUTPUT.PUT\_LINE('Item'|| LPAD('Unit Price',53,' ') || LPAD('Quantity',15,' ')|| LPAD('Line Total(RM)',19,' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

DBMS\_OUTPUT.PUT\_LINE('Treatment : '|| RPAD(v\_treatmenttype,35,' ') || LPAD(to\_char(v\_treatment\_price, '9999.99'),'43',' '));

IF transactionCount >0 THEN

FOR detail\_record IN detail\_cursor LOOP

select medic\_name, medic\_price into v\_medic\_name2, v\_medic\_price2

from medicalsupply

where medic\_id = detail\_record.medic\_id;

DBMS\_OUTPUT.PUT\_LINE(detail\_record.medic\_id||' : ' || RPAD(v\_medic\_name2,35,' ')||to\_char(v\_medic\_price2, '9999.99')||LPAD(detail\_record.line\_qty,15,' ')||LPAD(to\_char(detail\_record.line\_total, '9999.99'),20,' '));

END LOOP;

END IF;

DBMS\_OUTPUT.PUT\_LINE(in\_medic\_id || ' : ' || RPAD(v\_medic\_name,35,' ')|| to\_char(v\_medic\_price, '9999.99')||LPAD(in\_qty,15,' ')||LPAD(to\_char(v\_line\_total, '9999.99'),20,' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

insert into transactiondetail values(in\_transaction\_id,in\_medic\_id,in\_qty,v\_line\_total);

select total\_amount into v\_totalamount

from transaction where transaction\_id = in\_transaction\_id;

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE('Total Amount : '|| LPAD(to\_char(v\_totalamount, '9999.99'),'75',' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

EXCEPTION

WHEN e\_invalid\_transactionid THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

DBMS\_OUTPUT.PUT\_LINE ('Transaction ID not Found and Not Created yet !!!');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

WHEN e\_repeated\_medicid THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

DBMS\_OUTPUT.PUT\_LINE ('The Medic type already exist if you wish edit run edit procedure');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

WHEN e\_zero\_qty THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

DBMS\_OUTPUT.PUT\_LINE ('The quantity must be more than zero');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

WHEN e\_dayexceed THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

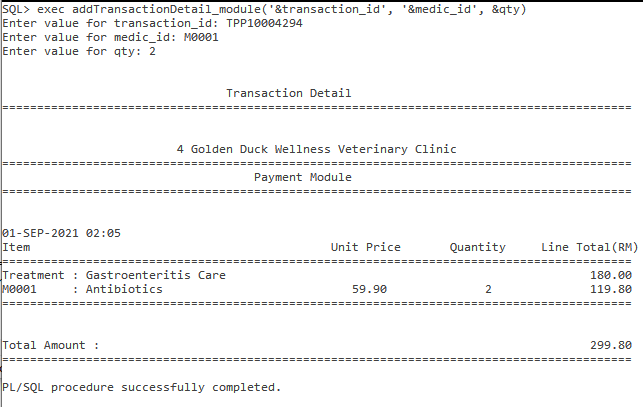
DBMS\_OUTPUT.PUT\_LINE ('Transaction that past 7 days cannot be edited');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

END;

/

#### Sample Output: After adding the total amount will be updated

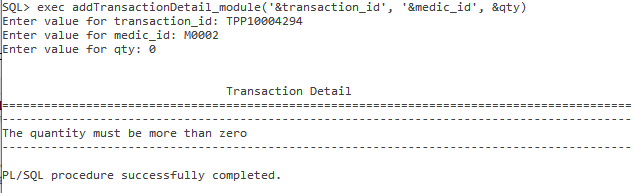


#### Exception output Output: When transaction id do not exist.

#### Exception output Output: When medic id entered it already exist in the transaction.



#### Exception output Output: When medic quantity less than 0.



**4.4.6 Procedure 3: Edit or delete transaction detail**

**Purpose: The purpose of this procedure is to allow the staff or veterinarian to edit or delete a transaction detail if any mistake entry is made. For example, extra medicine charges for the pet’s owner, or require more quantity of a medicine. The procedure will validate that the quantity must more than or equal 0. If the quantity is 0 means delete that transaction detail. If the transaction is a past 7 days transaction, the transaction details are not allowed to be edited. If the transaction detail entered did not exist an error will be prompted and any edition or deletion will update the transaction total amount.**

**Procedure code:**

CREATE OR REPLACE PROCEDURE editTransactionDetail\_module (in\_transaction\_id in TRANSACTION.transaction\_id%TYPE, in\_medic\_id in TRANSACTIONDETAIL.medic\_id%TYPE, in\_qty in TRANSACTIONDETAIL.line\_qty%TYPE) IS

transactionDetailCount NUMBER :=0;

medicCount NUMBER :=0;

v\_treatmenttype TREATMENT.treatment\_type%TYPE;

v\_treatment\_price TREATMENT.treatment\_price%TYPE;

v\_medic\_price MEDICALSUPPLY.medic\_price%TYPE;

v\_medic\_name MEDICALSUPPLY.medic\_name%TYPE;

v\_new\_line\_total TRANSACTIONDETAIL.line\_total%TYPE;

v\_line\_total TRANSACTIONDETAIL.line\_total%TYPE;

v\_line\_qty TRANSACTIONDETAIL.line\_qty%TYPE;

v\_transaction\_id TRANSACTIONDETAIL.transaction\_id%TYPE;

v\_transactiondate TRANSACTION.transaction\_datetime%TYPE;

v\_daydifferent number;

v\_samequantity number;

v\_totalamount TRANSACTION.total\_amount%TYPE;

e\_invalid\_transactiondetail EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_transactiondetail, -20158);

e\_lesszero\_qty EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_lesszero\_qty, -20159);

e\_day\_exceed EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_day\_exceed, -20160);

e\_samequantity EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_samequantity, -20161);

CURSOR detail\_cursor IS

SELECT \*

FROM transactiondetail

WHERE transaction\_id = in\_transaction\_id;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Transaction Detail',50, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

select count(\*)

INTO transactionDetailCount

FROM transactionDetail

WHERE transaction\_id = in\_transaction\_id AND medic\_id = in\_medic\_id ;

IF transactionDetailCount = 0 THEN

RAISE\_APPLICATION\_ERROR(-20158, 'Invalid Transaction Detail', true);

END IF;

IF in\_qty < 0 THEN

RAISE\_APPLICATION\_ERROR(-20159, 'Cannot Negative', true);

END IF;

SELECT SYSDATE - transaction\_datetime, transaction\_datetime INTO v\_daydifferent,v\_transactiondate

FROM DUAL, transaction

where transaction\_id = in\_transaction\_id ;

IF v\_daydifferent > 7 THEN

RAISE\_APPLICATION\_ERROR(-20160, 'Day Exceed', true);

END IF;

select count(\*)

INTO v\_samequantity

FROM transactionDetail

WHERE transaction\_id = in\_transaction\_id AND medic\_id = in\_medic\_id AND line\_qty = in\_qty ;

IF v\_samequantity > 0 THEN

RAISE\_APPLICATION\_ERROR(-20161, 'Same quantity', true);

END IF;

select line\_qty, line\_total

INTO v\_line\_qty, v\_line\_total

FROM transactionDetail

WHERE transaction\_id = in\_transaction\_id AND medic\_id = in\_medic\_id ;

select medic\_price, medic\_name

INTO v\_medic\_price, v\_medic\_name

FROM medicalsupply

WHERE medic\_id = in\_medic\_id;

select treatment\_type, treatment\_price into v\_treatmenttype, v\_treatment\_price

from transaction T, appointment A, Treatment R

where T.transaction\_id = in\_transaction\_id AND T.appointment\_id = A.appointment\_id AND A.treatment\_id = R.treatment\_id;

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('4 Golden Duck Wellness Veterinary Clinic',65, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Payment Module',50, ' '));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(TO\_CHAR(v\_transactiondate));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

DBMS\_OUTPUT.PUT\_LINE('Item'|| LPAD('Unit Price',53,' ') || LPAD('Quantity',15,' ')|| LPAD('Line Total(RM)',19,' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

DBMS\_OUTPUT.PUT\_LINE('Treatment : '|| RPAD(v\_treatmenttype,35,' ') || LPAD(to\_char(v\_treatment\_price, '9999.99'),'43',' '));

FOR detail\_record IN detail\_cursor LOOP

IF detail\_record.medic\_id != in\_medic\_id THEN

select medic\_name, medic\_price into v\_medic\_name, v\_medic\_price

from medicalsupply

where medic\_id = detail\_record.medic\_id;

DBMS\_OUTPUT.PUT\_LINE(detail\_record.medic\_id||' : ' || RPAD(v\_medic\_name,35,' ')||to\_char(v\_medic\_price, '9999.99')||LPAD(detail\_record.line\_qty,15,' ')||LPAD(to\_char(detail\_record.line\_total, '9999.99'),20,' '));

END IF;

END LOOP;

IF in\_qty = 0 THEN

DELETE FROM transactiondetail

WHERE transaction\_id = in\_transaction\_id AND medic\_id = in\_medic\_id;

ELSE

select medic\_name, medic\_price into v\_medic\_name, v\_medic\_price

from medicalsupply

where medic\_id = in\_medic\_id;

v\_new\_line\_total := in\_qty \* v\_medic\_price;

UPDATE TRANSACTIONDETAIL

SET line\_qty = in\_qty, line\_total = v\_new\_line\_total

WHERE transaction\_id = in\_transaction\_id AND medic\_id = in\_medic\_id;

DBMS\_OUTPUT.PUT\_LINE(in\_medic\_id||' : ' ||RPAD(v\_medic\_name,35,' ')||to\_char(v\_medic\_price, '9999.99')||LPAD(in\_qty,15,' ')||LPAD(to\_char(v\_new\_line\_total, '9999.99'),20,' '));

END IF;

select total\_amount into v\_totalamount

from transaction where transaction\_id = in\_transaction\_id;

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE('Total Amount : '|| LPAD(to\_char(v\_totalamount, '9999.99'),'75',' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

EXCEPTION

WHEN e\_invalid\_transactiondetail THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

DBMS\_OUTPUT.PUT\_LINE ('Invalid Transaction Detail and not FOUND !!!');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

WHEN e\_lesszero\_qty THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

DBMS\_OUTPUT.PUT\_LINE ('Quantity Cannot Less Than 0 !!!');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

WHEN e\_day\_exceed THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

DBMS\_OUTPUT.PUT\_LINE ('Transaction more than 7 days can be modify');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

WHEN e\_samequantity THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

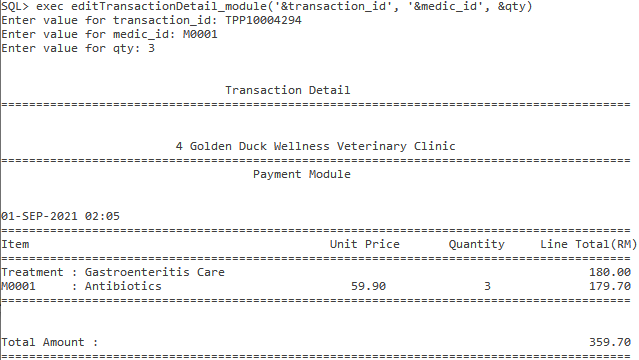
DBMS\_OUTPUT.PUT\_LINE ('Same as previous quantity no changes needed');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

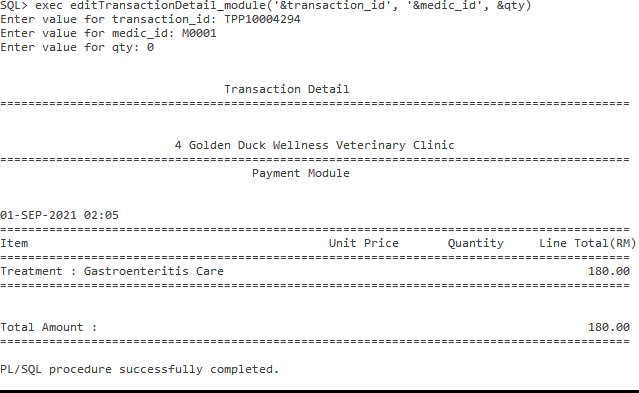
END;

/

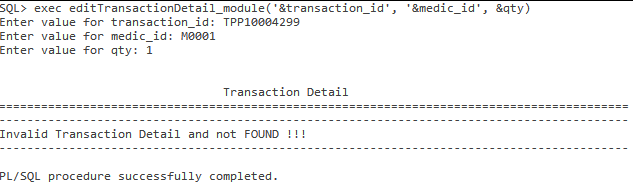
#### Sample Output: After editing existing transaction details, the total amount will be automatically updated.

****

#### Sample Output: After deleting existing transaction details, the total amount will be automatically updated.



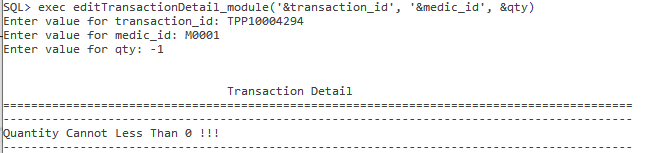
#### Exception Output: Entered transaction details that do not exist, an error message will be displayed.



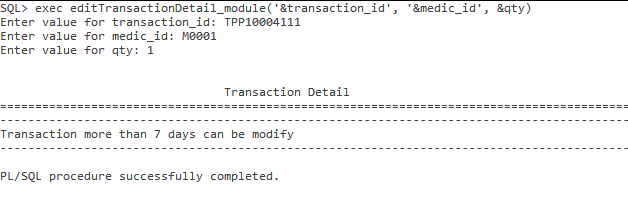
#### Exception Output: Past 7 days transaction cannot be edited.

#### 

#### Exception Output: Entered less than 0 , an error message will be displayed.



#### Exception Output: Past 7 days transactions are not allowed to be edited.



#### 

**4.4.7 Trigger 1: Validate Payment Date Time**

**Purpose: The purpose of this trigger is to validate payment date time. The pet’s owner is only able to make payment after the appointment. It is because the customer might cancel the appointment before the appointment date time. Thus, it only makes sense that the pet’s owner pays after the treatment for his or her pet has been done.**

**Trigger code:**

CREATE OR REPLACE TRIGGER trg\_paymentDateTime

BEFORE INSERT ON Transaction

FOR EACH ROW

DECLARE

v\_appointment\_date APPOINTMENT.appointment\_datetime%TYPE;

BEGIN

Select appointment\_datetime into v\_appointment\_date

FROM appointment

Where appointment\_id = :new.appointment\_id;

IF :new.transaction\_datetime<v\_appointment\_date THEN

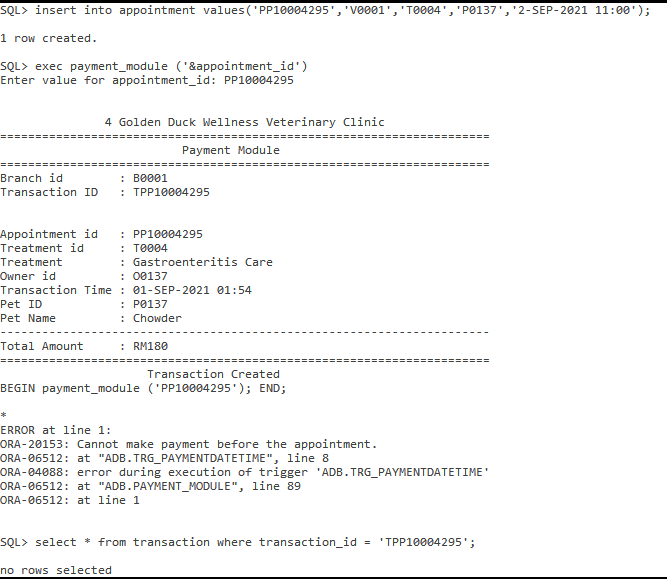
RAISE\_APPLICATION\_ERROR(-20153, 'Cannot make payment before the appointment.' );

END IF;

END;

/

**Sample Output:**

****

**4.4.8 Trigger 2: Update medicine quantity and transaction total amount after editing transaction detail**

**Purpose: The veterinarian might enter the transaction details wrongly. For example, the veterinarian charges the pet’s owner 3 Skin Care Lotions but the pet’s owner only buys 2 instead of 3. This allows the veterinarian to edit the quantity that is bought by the pet's owner. Same goes to if the pet’s owner wishes to buy even after entering the transaction details. After editing, ths trigger will auto update the stock quantity and the total amount of the transaction. Before is use in this case because before editing the system needs to make sure the stock quantity is enough before selling to the pet’s owner.**

**Trigger code:**

CREATE OR REPLACE TRIGGER TRG\_Update\_EditTransaction

BEFORE UPDATE ON TransactionDetail

FOR EACH ROW

DECLARE

v\_quantitydiff TRANSACTIONDETAIL.line\_qty%TYPE;

v\_newqty TRANSACTIONDETAIL.line\_qty%TYPE;

v\_newlinetotal TRANSACTIONDETAIL.line\_total%TYPE;

v\_medicprice MEDICALSUPPLY.medic\_price%TYPE;

v\_linetotal\_dif TRANSACTIONDETAIL.line\_total%TYPE;

BEGIN

Select medic\_price into v\_medicprice

from medicalsupply where medic\_id = :old.medic\_id;

v\_newlinetotal := :new.line\_total;

IF :new.line\_qty > :old.line\_qty THEN

Update medicalsupply

SET medic\_qty = medic\_qty - (:new.line\_qty - :old.line\_qty)

where medic\_id = :old.medic\_id;

v\_linetotal\_dif := v\_newlinetotal - :old.line\_total;

Update Transaction

SET total\_amount = total\_amount + v\_linetotal\_dif

where transaction\_id = :old.transaction\_id;

ELSE

Update medicalsupply

SET medic\_qty = medic\_qty + (:old.line\_qty - :new.line\_qty )

where medic\_id = :old.medic\_id;

v\_linetotal\_dif := :old.line\_total - v\_newlinetotal;

Update Transaction

SET total\_amount = total\_amount - v\_linetotal\_dif

where transaction\_id = :old.transaction\_id;

END IF;

END;/

**4.4.6 Trigger 3: Update the stock and transaction total amount after deleting transaction detail**

**Purpose: The purpose of this trigger is to Update the stock and transaction total amount after deleting transaction details. The veterinarian might accidentally charge the pet’s owner with medicine that the pet owner didn't purchase. Hence, after the pet’s owner or the veterinarian noticed the mistake. The veterinarian will delete that transaction detail. Thus this trigger will automatically the stock quantity and the total amount of the transaction.**

**Trigger code:**

CREATE OR REPLACE TRIGGER TRG\_Update\_DeleteTransaction

BEFORE DELETE ON TransactionDetail

FOR EACH ROW

BEGIN

Update Transaction

SET total\_amount = total\_amount - :old.line\_total

where transaction\_id = :old.transaction\_id;

Update MedicalSupply

SET medic\_qty = medic\_qty + :old.line\_qty

where medic\_id = :old.medic\_id;

END;

/

**4.4.7 Trigger 4: Update transaction total amount**

**Purpose: The purpose of this trigger is to keep adding the transaction amount after the pet’s owner keep buying new medicine for his or her pet after the treatment.**

**Trigger code:**

CREATE OR REPLACE TRIGGER TRG\_Update\_Total\_Amount

After Insert ON TransactionDetail

FOR EACH ROW

BEGIN

Update Transaction

SET total\_amount = total\_amount + :new.line\_total

where transaction\_id = :new.transaction\_id;

END;

/

**4.4.11 Report 1: Summary report of Veterinarian Performance in a month.**

**Purpose: The purpose of this report is to monitor the performance of a veterinarian in a month and how much revenue the veterinarian brings to the company in a specific month. The number of types of treatments that are handled by the veterinarian will also be shown.**

SQL statement:

CREATE OR REPLACE PROCEDURE RPT\_Revenue\_Vet(IN\_vetid IN VETERINARIAN.vet\_id%TYPE ,IN\_year IN NUMBER, IN\_month IN NUMBER) IS

v\_vetcount NUMBER := 0;

v\_treatmentcount NUMBER := 0;

v\_vetID VETERINARIAN.vet\_id%TYPE;

v\_vetname VETERINARIAN.vet\_name%TYPE;

v\_vetcontact VETERINARIAN.vet\_contact%TYPE;

v\_branchid BRANCH.branch\_id%TYPE;

v\_branchS BRANCH.state%TYPE;

v\_branchC BRANCH.city%TYPE;

v\_branchP BRANCH.postcode%TYPE;

v\_branchST BRANCH.streetname%TYPE;

v\_sumQuantity NUMBER(10,2) := 0;

v\_sumRevenue NUMBER(10,2) := 0;

v\_month VARCHAR2(11);

v\_maxYear NUMBER(4);

v\_minYear NUMBER(4);

v\_maxMonth NUMBER;

v\_sysdate DATE;

e\_invalid\_vetid EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_vetid, -20162);

e\_invalid\_month EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_month, -20163);

e\_invalid\_year EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_year, -20164);

CURSOR Treatment\_CURSOR IS

SELECT T.treatment\_id, T.treatment\_type, count(A.appointment\_id) AS Number\_of\_Treatment,

t.treatment\_price, (count(A.appointment\_id)\*T.treatment\_price) AS

TotalAmount

FROM Appointment A, Treatment T, Veterinarian V

WHERE A.treatment\_id = T.treatment\_id AND V.vet\_id = A.vet\_id

AND EXTRACT(YEAR FROM appointment\_datetime) = IN\_year

AND EXTRACT(MONTH FROM appointment\_datetime) = IN\_month

AND A.vet\_id = IN\_vetid

GROUP BY T.treatment\_id, T.treatment\_type, T.treatment\_price

ORDER BY TotalAmount ASC;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('4 Golden Duck Wellness Veterinary Clinic',65, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Vet Summary Report based on the revenue of each treatment',70, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

Select count(V.vet\_id) INTO v\_vetcount from veterinarian V Where V.vet\_id = IN\_vetid;

IF v\_vetcount = 0 THEN

RAISE\_APPLICATION\_ERROR(-20162, 'Invalid Vet ID', true);

END IF;

IF IN\_MONTH < 1 OR IN\_MONTH > 12 THEN

RAISE\_APPLICATION\_ERROR(-20163, 'Invalid Month', true);

END IF;

Select V.vet\_id, V.vet\_name,V.vet\_contact,B.branch\_id, B.state, B.city, B.Postcode,B.streetname,

Extract(Year FROM Max(A.appointment\_datetime)),Extract(Year FROM Min(A.appointment\_datetime))

INTO v\_vetID, v\_vetname, v\_vetcontact, v\_branchid, v\_branchS, v\_branchC, v\_branchP,v\_branchST,

v\_maxyear, v\_minyear

From veterinarian V, branch B, Appointment A

Where A.vet\_id = V.vet\_id AND V.branch\_id = B.branch\_id AND V.vet\_id = IN\_vetid

group by V.vet\_id, V.vet\_name,V.vet\_contact,B.branch\_id, B.state, B.city, B.Postcode,B.streetname;

IF IN\_YEAR < v\_minyear OR IN\_YEAR > v\_maxyear THEN

RAISE\_APPLICATION\_ERROR(-20164, 'Invalid Year', true);

END IF;

Select sysdate into v\_sysdate from dual;

DBMS\_OUTPUT.PUT\_LINE('Report Generated on : ' || v\_sysdate);

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE('Veterinarian ID : ' || v\_vetID);

DBMS\_OUTPUT.PUT\_LINE('Veterinarian Name : ' || v\_vetName);

DBMS\_OUTPUT.PUT\_LINE('Veterinarian Contact : ' || v\_vetContact);

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE('Branch ID : ' || v\_branchid);

DBMS\_OUTPUT.PUT\_LINE('Branch State : ' || v\_branchS);

DBMS\_OUTPUT.PUT\_LINE('Branch City : ' || v\_branchC);

DBMS\_OUTPUT.PUT\_LINE('Branch Postcode : ' || v\_branchP);

DBMS\_OUTPUT.PUT\_LINE('Branch Street : ' || v\_branchS);

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

SELECT TO\_CHAR(TO\_DATE(IN\_MONTH, 'MM'), 'MONTH') INTO v\_month FROM DUAL;

DBMS\_OUTPUT.PUT\_LINE(LPAD('Year ' || IN\_YEAR || ' '|| v\_month,55,' '));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Treatment ID',15,' ')|| ' | ' || RPAD('Treatment Type',25,' ')|| ' | ' ||LPAD('No',2,' ')|| ' | ' ||LPAD('Treatment Price (RM)',20,' ')|| ' | ' ||LPAD('Total (RM)',15,' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

FOR treatment\_record IN Treatment\_CURSOR LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD(treatment\_record.treatment\_id, 15, ' ')|| ' | ' || RPAD(treatment\_record.treatment\_type,25,' ')|| ' | ' || LPAD(treatment\_record.Number\_of\_Treatment,2,' ')

|| ' | ' || LPAD(TO\_CHAR(treatment\_record.treatment\_price, 999.99),20,' ')|| ' | ' || LPAD(to\_char(treatment\_record.TotalAmount, '999999.99'),15,' '));

v\_sumQuantity := v\_sumQuantity + treatment\_record.Number\_of\_Treatment;

v\_sumRevenue := v\_sumRevenue + treatment\_record.TotalAmount;

v\_treatmentcount := v\_treatmentcount + 1;

END LOOP;

IF v\_treatmentcount = 0 THEN

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('No treatment handled this month !!!',60,' '));

END IF;

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

DBMS\_OUTPUT.PUT\_LINE('Total Treatment : '|| v\_sumQuantity);

DBMS\_OUTPUT.PUT\_LINE('Total Treatment Revenue : RM'|| to\_char(v\_sumRevenue,99999.99));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',90,'='));

EXCEPTION

WHEN e\_invalid\_vetid THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

DBMS\_OUTPUT.PUT\_LINE ('Vet ID Does Not Exist !!!');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

WHEN e\_invalid\_month THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

DBMS\_OUTPUT.PUT\_LINE ('Invalid Month must be within (1-12)');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

WHEN e\_invalid\_year THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

DBMS\_OUTPUT.PUT\_LINE ('Invalid Year or No Treatment by '|| v\_vetid || ' : ' || v\_vetname || ' in this year');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',90,'-'));

END;

/

alter session set nls\_date\_format = 'DD-MON-YYYY';

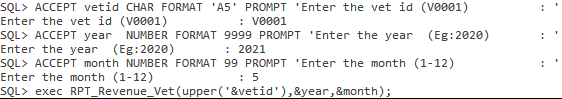
ACCEPT vetid CHAR FORMAT 'A5' PROMPT 'Enter the vet id (V0001) : '

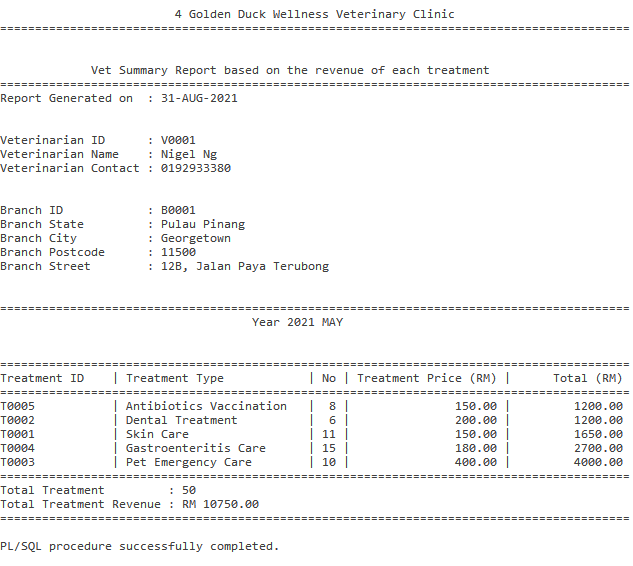
ACCEPT year NUMBER FORMAT 9999 PROMPT 'Enter the year (Eg:2020) : '

ACCEPT month NUMBER FORMAT 99 PROMPT 'Enter the month (1-12) : '

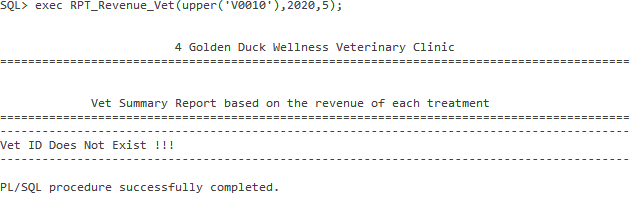
exec RPT\_Revenue\_Vet(upper('&vetid'),&year,&month);

**Sample Output:**

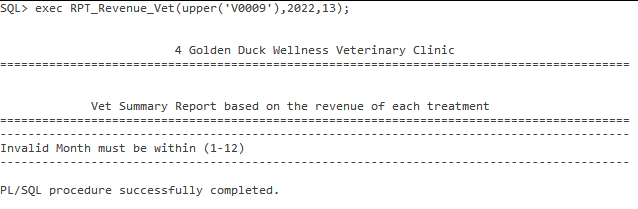




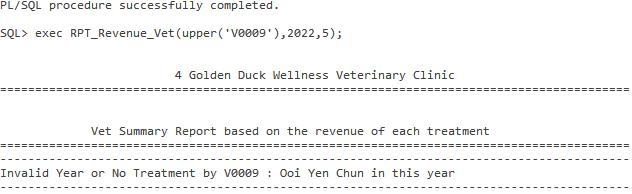
Exception Output : Invalid veterinarian ID



Exception Output : Invalid Month



Exception Output : Invalid year or no treatment for the veterinarian in that month and year



**4.4.9 Report 2: On demand report of days that performed poorly for a specific with a date range**

**Purpose: The purpose of this report is to analyze days that performed poorly within a date range. For example, Branch “B0001” from 1-MAY-2021 to 31-MAY-2021 which days that the brunch has revenue lower than RM 2000. Then a list of days with all the transactions will be displayed. At the end of the report will show which day has the revenue that is less than RM 2000. For example, within the date range two days that have lower revenue are on revenue. This allows the branch head to know which specific day has lower revenue and is required to improve it.**

SQL statement:

CREATE OR REPLACE PROCEDURE RPT\_Less\_Revenue(IN\_branchID IN Transaction.branch\_id%TYPE ,IN\_StartDate in DATE, IN\_EndDate in DATE, IN\_MinAmount in Number) IS

CURSOR less\_transaction\_CURSOR IS

select trunc(transaction\_datetime) as TransactionDate,to\_char(transaction\_datetime, 'DAY') AS Day,

count(transaction\_id) AS TotalTransaction,

sum(total\_amount) AS Total\_Amount, ((sum(total\_amount))/(count(transaction\_id))) AS AverageT

from transaction

where transaction\_datetime between IN\_StartDate AND IN\_EndDate AND branch\_id = IN\_branchID

Having sum(total\_amount) < IN\_MinAmount

group by trunc(transaction\_datetime), to\_char(transaction\_datetime, 'DAY')

order by 1;

CURSOR transactiondetail\_CURSOR (v\_transaction\_datetime DATE) IS

select T.transaction\_id , T.transaction\_datetime, TT.treatment\_id, TT.treatment\_type, TT.treatment\_price,

TD.medic\_id, M.medic\_name, TD.line\_qty,M.medic\_price, TD.line\_total

from transaction T, appointment A, treatment TT, transactiondetail TD, medicalsupply M

where T.appointment\_id = A.appointment\_id AND A.treatment\_id = TT.treatment\_id AND

TD.transaction\_id = T.transaction\_id AND TD.medic\_id = M.medic\_id AND

trunc(T.transaction\_datetime) = v\_transaction\_datetime AND T.branch\_id = IN\_branchID

order by 1;

CURSOR less\_day\_CURSOR IS

select to\_char(DateTime, 'DAY') AS Day, count(\*) AS TotalDay

from (select trunc(transaction\_datetime) AS DateTime, sum(total\_amount) as Total\_amount

from Transaction

Where transaction\_datetime Between '01-MAY-2021' AND '31-MAY-2021' AND branch\_id = 'B0001'

group by trunc(transaction\_datetime)

having sum(total\_amount) < 2000)

where Total\_amount < 3000

group by to\_char(DateTime, 'DAY')

order by 2;

v\_totalCount NUMBER := 0;

v\_detailCount NUMBER := 0;

v\_transactionid TRANSACTION.transaction\_id%type := 'TPP00000000';

v\_rowCount NUMBER := 0;

v\_sysdate DATE;

v\_datecount NUMBER :=0;

v\_branchcount NUMBER :=0;

e\_invalid\_branch EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_branch, -20165);

BEGIN

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('4 Golden Duck Wellness Veterinary Clinic',95, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',150,'='));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('On Demand Report', 85, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Days that have less revenue RM'||to\_char(IN\_MinAmount,999999.99),95, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('FROM '|| IN\_StartDate ||' to '|| IN\_EndDate,90, ' '));

Select sysdate into v\_sysdate from dual;

DBMS\_OUTPUT.PUT\_LINE('Report Generated on : ' || v\_sysdate);

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',150,'='));

Select count(\*) into v\_branchcount from branch where branch\_id = IN\_branchID;

IF v\_branchcount = 0 THEN

RAISE\_APPLICATION\_ERROR(-20165, 'Invalid Branch ID', true);

END IF;

FOR less\_transaction\_record IN less\_transaction\_CURSOR LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD('DATE : ' || less\_transaction\_record.TransactionDate,23, ' ')||

LPAD('| DAY : ' || less\_transaction\_record.Day,20, ' ') ||

LPAD('| No Transaction :' || to\_char(less\_transaction\_record.TotalTransaction,99),25,' ') ||

LPAD('| Total Amount RM '|| to\_char(less\_transaction\_record.Total\_Amount,9999.99),30,' ')||

LPAD('| Average Amount RM '|| to\_char(less\_transaction\_record.AverageT,999.99),30,' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',150,'='));

FOR transactiondetail\_record IN transactiondetail\_CURSOR (less\_transaction\_record.TransactionDate) LOOP

IF v\_transactionid != transactiondetail\_record.transaction\_id THEN

DBMS\_OUTPUT.PUT\_LINE(chr(10));

v\_transactionid := transactiondetail\_record.transaction\_id;

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',150,'-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Transaction ID',15, ' ')|| ' | ' ||RPAD('Treatment ID',12, ' ')|| ' | ' ||

LPAD('Treatment Type',25, ' ')|| ' | ' ||LPAD('Treatment Price',15, ' ')|| ' | ' ||

LPAD('Medicine ID',12, ' ')|| ' | ' || LPAD('Medicine Name',17, ' ')|| ' | ' ||

LPAD('QTY',3, ' ')|| ' | ' ||LPAD('Medicine Price',15, ' ')|| ' | ' ||LPAD('Total (RM)',10, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',150,'-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD(transactiondetail\_record.Transaction\_id,15, ' ')||' | ' || RPAD(transactiondetail\_record.Treatment\_id,12, ' ')||' | ' ||

LPAD(transactiondetail\_record.treatment\_type,25, ' ')||' | ' ||LPAD(to\_char(transactiondetail\_record.treatment\_price, 999.99),15, ' ')||' | '||

LPAD(transactiondetail\_record.medic\_id,12, ' ')||' | ' || LPAD(transactiondetail\_record.medic\_name,17, ' ')||' | ' ||

LPAD(to\_char(transactiondetail\_record.line\_qty,99),3, ' ') ||' | ' || LPAD(to\_char(transactiondetail\_record.medic\_price, 999.99),15, ' ') ||' | ' ||

LPAD(to\_char(transactiondetail\_record.line\_total,9999.99),10, ' '));

ELSE

DBMS\_OUTPUT.PUT\_LINE(LPAD(transactiondetail\_record.medic\_id,91, ' ')|| ' | ' || LPAD(transactiondetail\_record.medic\_name,17, ' ')|| ' | ' ||

LPAD(to\_char(transactiondetail\_record.line\_qty,99),3, ' ')||' | '|| LPAD(to\_char(transactiondetail\_record.medic\_price, 999.99),15, ' ')|| ' | ' ||

LPAD(to\_char(transactiondetail\_record.line\_total,9999.99),10, ' '));

END IF;

v\_rowCount := v\_detailcount + 1;

END LOOP;

v\_totalcount := v\_totalcount +1;

DBMS\_OUTPUT.PUT\_LINE(chr(10)||chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',150,'='));

END LOOP;

IF v\_rowCount = 0 THEN

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('No sales than less RM '|| to\_char(IN\_MinAmount,99999.99) ||'than during this period',70, ' '));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

ELSE

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',50,'='));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Count of days from MONDAY to SUNDAY',50,' ')|| '|');

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',50,'='));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Day',20,' ')||' | ' ||RPAD('Number',27,' ') || '|');

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',50,'-'));

FOR dayCount\_record in less\_day\_CURSOR LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD(dayCount\_record.Day,20, ' ')|| ' | ' ||RPAD(dayCount\_record.TotalDay,27, ' ')|| '|');

v\_dateCount := v\_dateCount + dayCount\_record.TotalDay;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',50,'='));

DBMS\_OUTPUT.PUT\_LINE('Total ' || v\_dateCount || ' days Revenue Less Than RM' || to\_char(IN\_MinAmount,99999.99));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',50,'='));

END IF;

EXCEPTION

WHEN e\_invalid\_branch THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',150,'-'));

DBMS\_OUTPUT.PUT\_LINE ('Branch ID Does Not Exist !!!');

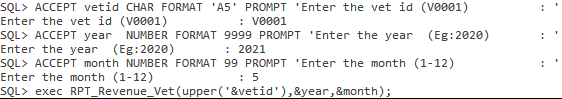
DBMS\_OUTPUT.PUT\_LINE(LPAD('-',150,'-'));

END;

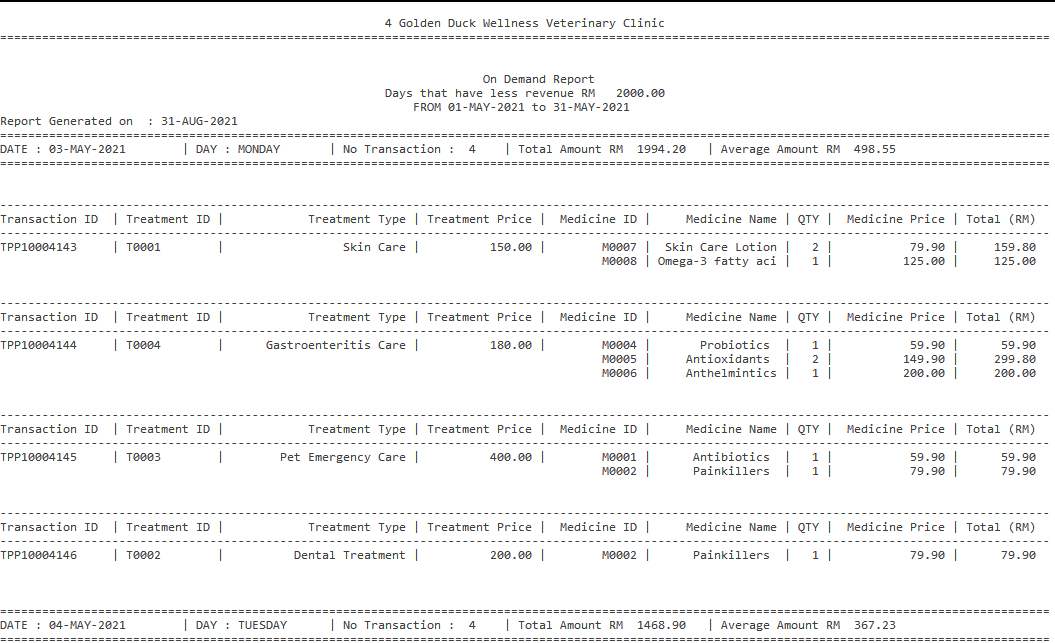
/

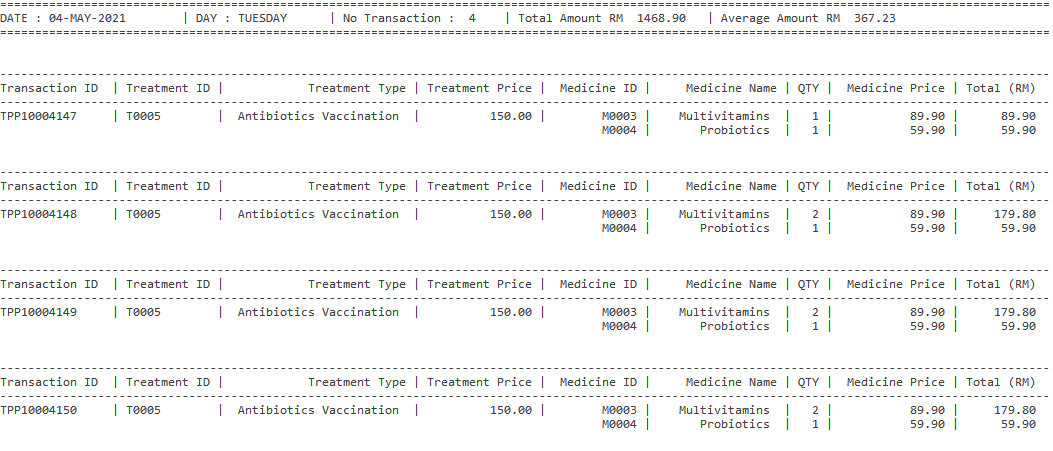
**Exception Output: Out of clinic operated year.**

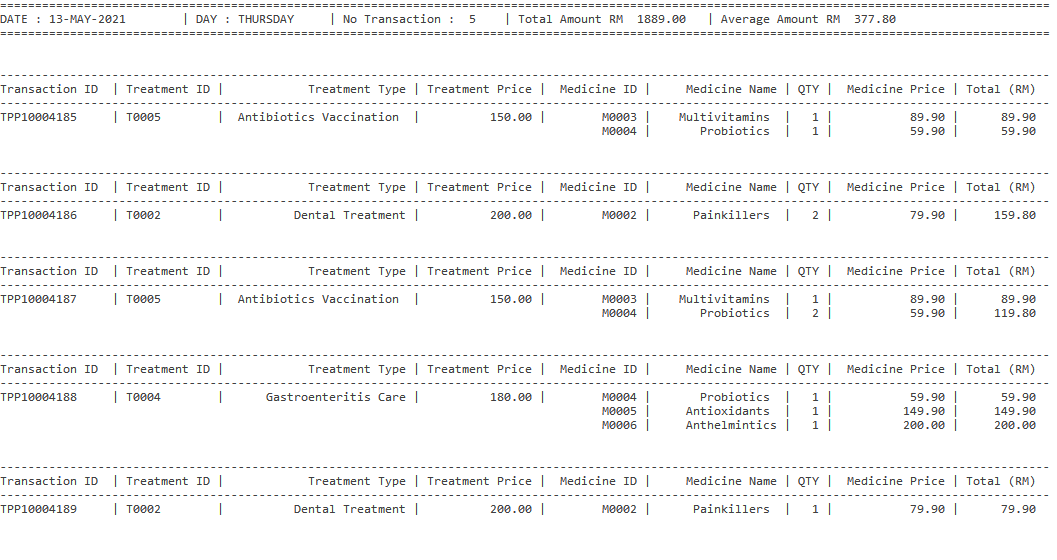
**Sample Output:**

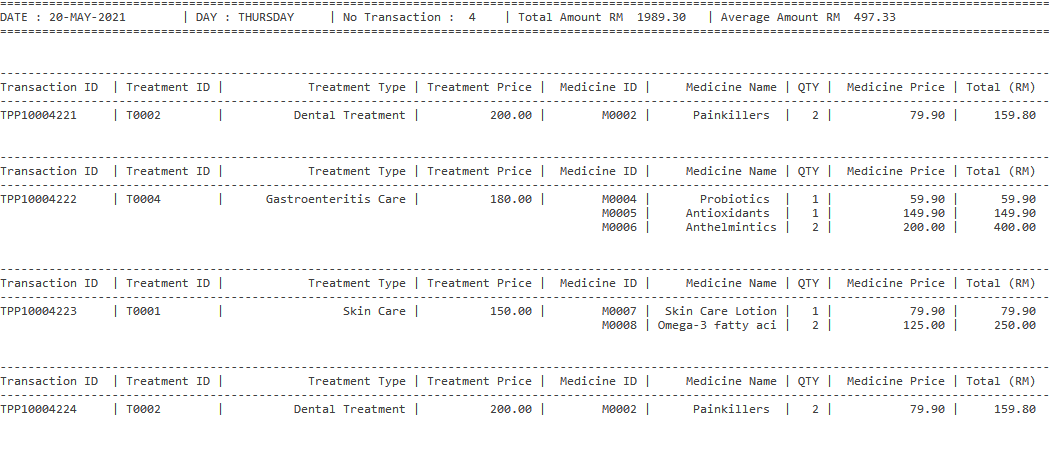


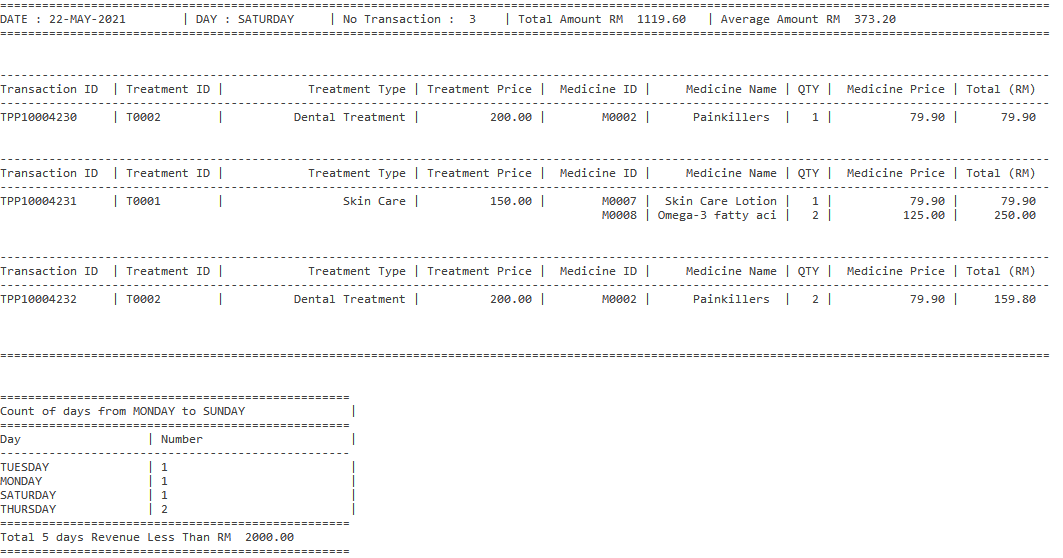
Sample Output :



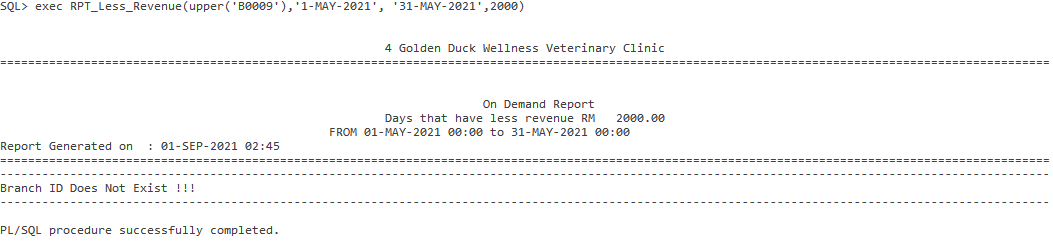








**Exception Output: Invalid Branch ID**

****

**4.4.10 Report 3: Detail report of each Branch’s Performance in a specific year**

**Purpose: The purpose of this report is to show total revenue, revenue of each treatment and revenue of each medecine of a branch. Details such as treatment done, medicine sold by each branch will also be shown, contribution of each treatment and medicine to the branch revenue will also be shown . The contribution of each branch to the total revenue of the specific year will be shown in percentage.**

SQL statement:

CREATE OR REPLACE PROCEDURE RPT\_Branch\_Performance(in\_YEAR IN NUMBER) IS

CURSOR branch\_overall\_revenue\_CURSOR IS

select B.branch\_id , B.state, B.city, B.postcode, B.streetname,

count(T.transaction\_id) AS TotalTransaction,

sum(T.total\_amount) AS Total\_Amount

from transaction T, Branch B

where T.branch\_id = B.branch\_id AND Extract(YEAR from transaction\_datetime) = 2020

group by B.branch\_id , B.state, B.city, B.postcode, B.streetname

order by 1;

CURSOR branch\_treatment\_revenue\_CURSOR (v\_branchID BRANCH.branch\_id%TYPE) IS

select B.branch\_id , TT.treatment\_id, TT.treatment\_type, TT.treatment\_price,

count(T.transaction\_id) as TotalTreatment,

(count(T.transaction\_id)\*treatment\_price) AS TotalTreatmentRevenue

from transaction T, Branch B, appointment A, treatment TT

where T.branch\_id = B.branch\_id AND T.appointment\_id = A.appointment\_id AND

A.treatment\_id = TT.treatment\_id AND

Extract(YEAR from transaction\_datetime) = in\_YEAR AND

T.branch\_id = v\_branchID

group by B.branch\_id, TT.treatment\_id, TT.treatment\_type, TT.treatment\_price

order by 1,2;

CURSOR branch\_medic\_revenue\_CURSOR (v\_branchID BRANCH.branch\_id%TYPE) IS

select T.branch\_id , M.medic\_id, M.medic\_name, M.medic\_price,

sum(TD.line\_qty) as TotalMedic,

sum(TD.line\_total) AS TotalMedicRevenue

from transaction T, transactiondetail TD, medicalsupply M

where T.transaction\_id = TD.transaction\_id AND TD.medic\_id = M.medic\_id AND

Extract(YEAR from T.transaction\_datetime) = in\_YEAR AND

T.branch\_id = v\_branchID

group by T.branch\_id, M.medic\_id, M.medic\_name, M.medic\_price

order by 1,2;

v\_branchTotalTreatmentRevenue NUMBER := 0;

v\_branchTotalMedicRevenue NUMBER := 0;

v\_all\_totalamount NUMBER;

v\_all\_totalTreatment NUMBER :=0;

v\_all\_totalMedical NUMBER :=0;

v\_transactionid TRANSACTION.transaction\_id%type := 'TPP00000000';

v\_rowCount NUMBER := 0;

v\_sysdate DATE;

v\_branchcount NUMBER :=0;

v\_allRevenue NUMBER :=0;

v\_minYear NUMBER;

v\_maxYear NUMBER;

e\_invalid\_year EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_year, -20166);

BEGIN

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('4 Golden Duck Wellness Veterinary Clinic',80, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',115,'='));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Detail Report', 63, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Each Branch Performance Report',72, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD(IN\_YEAR,58, ' '));

Select sysdate into v\_sysdate from dual;

DBMS\_OUTPUT.PUT\_LINE('Report Generated on : ' || v\_sysdate);

Select Extract(YEAR FROM Min(transaction\_datetime)), Extract(YEAR FROM Max(transaction\_datetime))

into v\_minYear, v\_maxYear

From transaction;

IF in\_YEAR < v\_minYear or in\_Year > v\_maxYear THEN

RAISE\_APPLICATION\_ERROR(-20166, 'Invalid Year', true);

END IF;

FOR branch\_record IN branch\_overall\_revenue\_CURSOR LOOP

DBMS\_OUTPUT.PUT\_LINE(LPAD('=',115,'='));

DBMS\_OUTPUT.PUT\_LINE('Branch ID :' || branch\_record.branch\_id);

DBMS\_OUTPUT.PUT\_LINE('State :' || branch\_record.state);

DBMS\_OUTPUT.PUT\_LINE('City :' || branch\_record.city);

DBMS\_OUTPUT.PUT\_LINE('Postcode :' || branch\_record.postcode);

DBMS\_OUTPUT.PUT\_LINE('Street Name :' || branch\_record.streetname);

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Revenue for each treatment',75,' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Treatment ID', 12 ,' ')|| ' | ' || RPAD('Treatment Type', 20 ,' ')|| ' | ' ||

LPAD('Treatment Price (RM)', 20 ,' ')|| ' | ' ||

LPAD('No Treatment', 12 ,' ')|| ' | ' ||

LPAD('Treatment Revenue (RM)', 25 ,' ')|| ' | ' ||

LPAD('Percent%',10,' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

FOR treatment\_record1 IN branch\_treatment\_revenue\_CURSOR (branch\_record.branch\_id) LOOP

v\_branchTotalTreatmentRevenue := v\_branchTotalTreatmentRevenue + treatment\_record1.TotalTreatmentRevenue;

END LOOP;

FOR treatment\_record IN branch\_treatment\_revenue\_CURSOR (branch\_record.branch\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD(treatment\_record.treatment\_id, 12 ,' ')|| ' | ' ||

RPAD(treatment\_record.treatment\_type, 20 ,' ')|| ' | ' ||

LPAD(to\_char(treatment\_record.treatment\_price, 999.99), 20 ,' ')|| ' | ' ||

LPAD(treatment\_record.TotalTreatment, 12 ,' ')|| ' | ' ||

LPAD(to\_char(treatment\_record.TotalTreatmentRevenue, 999999.99), 25 ,' ')|| ' | ' ||

LPAD(round((treatment\_record.TotalTreatmentRevenue/v\_branchTotalTreatmentRevenue\*100),2),10,' '));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

DBMS\_OUTPUT.PUT\_LINE('Total (RM)'||LPAD(to\_char(v\_branchTotalTreatmentRevenue,99999999.99),91,' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Revenue for each Medicine',75,' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Medicine ID', 12 ,' ')|| ' | ' || RPAD('Medicine Name', 20 ,' ')|| ' | ' ||

LPAD('Medicine Price (RM)', 20 ,' ')|| ' | ' ||

LPAD('QTY Sold', 12 ,' ')|| ' | ' ||

LPAD('Medicine Revenue (RM)', 25 ,' ')|| ' | ' ||

LPAD('Percent%',10,' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

FOR medic\_record1 IN branch\_medic\_revenue\_CURSOR (branch\_record.branch\_id) LOOP

v\_branchTotalMedicRevenue := v\_branchTotalMedicRevenue + medic\_record1.TotalMedicRevenue;

END LOOP;

FOR medic\_record IN branch\_medic\_revenue\_CURSOR (branch\_record.branch\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD(medic\_record.medic\_id, 12 ,' ')|| ' | ' ||

RPAD(medic\_record.medic\_name, 20 ,' ')|| ' | ' ||

LPAD(to\_char(medic\_record.medic\_price, 999.99), 20 ,' ')|| ' | ' ||

LPAD(medic\_record.TotalMedic, 12 ,' ')|| ' | ' ||

LPAD(to\_char(medic\_record.TotalMedicRevenue, 999999.99), 25 ,' ')|| ' | ' ||

LPAD(round((medic\_record.TotalMedicRevenue/v\_branchTotalMedicRevenue\*100),2),10,' '));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

DBMS\_OUTPUT.PUT\_LINE('Total (RM)'||LPAD(to\_char(v\_branchTotalMedicRevenue,99999999.99),91,' '));

v\_branchTotalTreatmentRevenue:= 0;

v\_branchTotalMedicRevenue:= 0;

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

DBMS\_OUTPUT.PUT\_LINE(chr(10));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

DBMS\_OUTPUT.PUT\_LINE(LPAD('Contribution of each branch in ',70,' ' )|| IN\_YEAR);

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Branch ID ', 10, ' ') ||' | ' || LPAD('Treatment Revenue (RM)',25,' ') || ' | ' ||

LPAD('Medicine Revenue (RM)',25,' ') ||' | ' || LPAD('Branch Total Revenue (RM)',25,' ') ||' | ' || LPAD('Contribution%',17,' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

Select sum(total\_amount) into v\_all\_totalamount from transaction where Extract(YEAR from transaction\_datetime) = IN\_year;

FOR branch\_record IN branch\_overall\_revenue\_CURSOR LOOP

v\_branchTotalTreatmentRevenue:= 0;

v\_branchTotalMedicRevenue:= 0;

FOR treatment\_record1 IN branch\_treatment\_revenue\_CURSOR (branch\_record.branch\_id) LOOP

v\_branchTotalTreatmentRevenue := v\_branchTotalTreatmentRevenue + treatment\_record1.TotalTreatmentRevenue;

END LOOP;

FOR medic\_record1 IN branch\_medic\_revenue\_CURSOR (branch\_record.branch\_id) LOOP

v\_branchTotalMedicRevenue := v\_branchTotalMedicRevenue + medic\_record1.TotalMedicRevenue;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(RPAD(branch\_record.branch\_id, 10, ' ') ||' | ' ||

LPAD(to\_char(v\_branchTotalTreatmentRevenue,999999.99),25,' ') || ' | ' ||

LPAD(to\_char(v\_branchTotalMedicRevenue, 999999.99),25, ' ') ||' | ' ||

LPAD(to\_char(branch\_record.Total\_Amount, 999999.99),25,' ') ||' | ' ||

LPAD(round((branch\_record.Total\_Amount/v\_all\_totalamount\*100),2),17,' '));

v\_all\_totalTreatment := v\_all\_totalTreatment + v\_branchTotalTreatmentRevenue;

v\_all\_totalMedical := v\_all\_totalMedical + v\_branchTotalMedicRevenue;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

DBMS\_OUTPUT.PUT\_LINE('Total (RM) |'|| LPAD(to\_char(v\_all\_totalTreatment,9999999.99),26,' ') ||' | ' ||

LPAD(to\_char(v\_all\_totalMedical,9999999.99),25,' ')||' | ' ||

LPAD(to\_char( v\_all\_totalamount,9999999.99),25,' ')||' | ' );

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

EXCEPTION

WHEN e\_invalid\_year THEN

DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

DBMS\_OUTPUT.PUT\_LINE ('The year you have entered is invalid or within the range of operated year !!!');

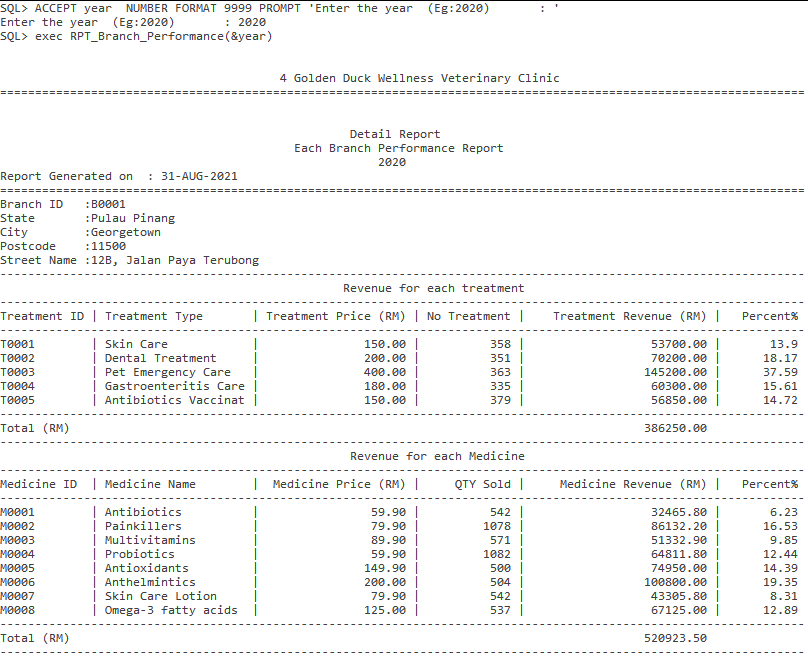
DBMS\_OUTPUT.PUT\_LINE(LPAD('-',115,'-'));

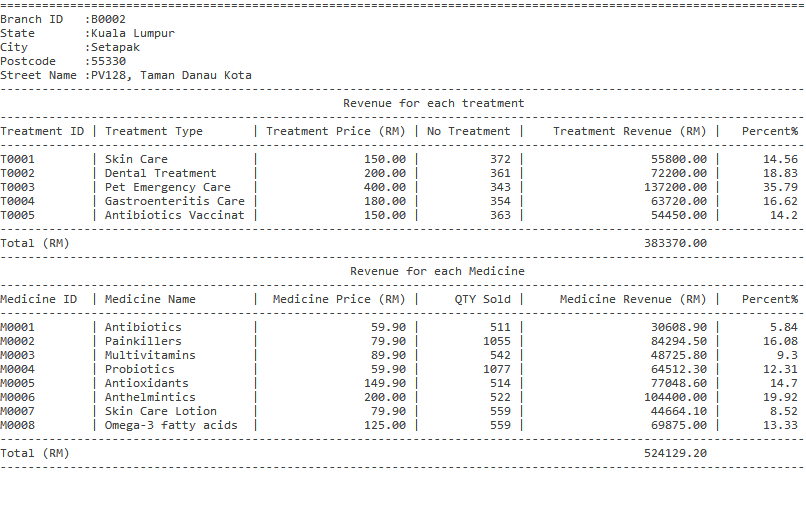
END ;

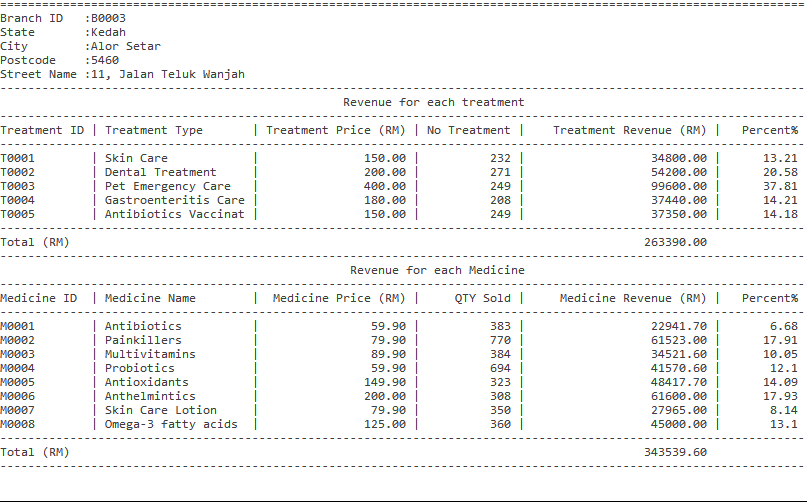
/

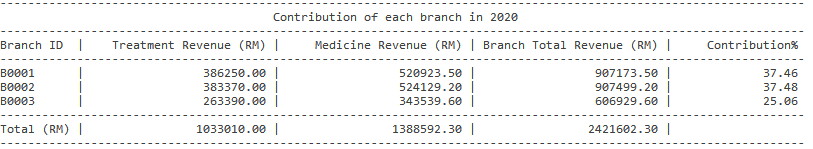
exec RPT\_Branch\_Performance(&year)

**Sample Output :**

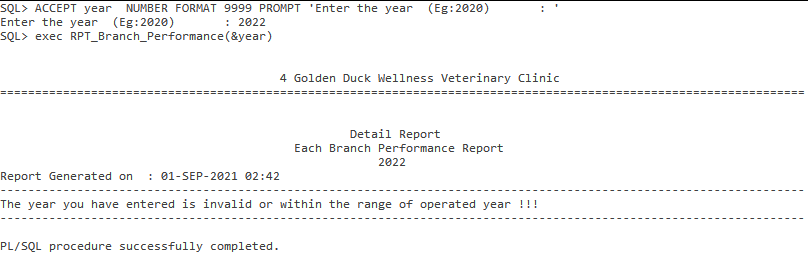








**Exception Output : Invalid year and out of clinic operation year**

****

### Chapter 5 Extra Effort Highlights

**5.1 (Tan Yi Hong)**

**5.1.1 Views**

**View 1: The purpose of this view is to display the top pet type that received treatment in each branch**

CREATE OR REPLACE VIEW topPetTreatment AS

SELECT t.branch\_id, pt.type\_name, COUNT(t.appointment\_id) AS NoOfTreatment, SUM(t.total\_amount) AS TransactionAmount

FROM appointment a, veterinarian v, pet p, petType pt, transaction t

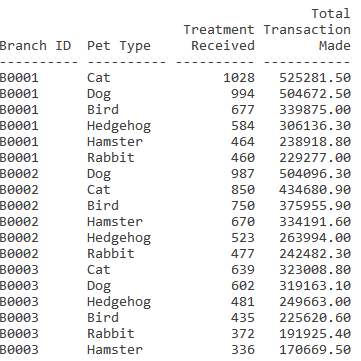
WHERE t.appointment\_id=a.appointment\_id AND a.vet\_id=v.vet\_id AND a.pet\_id=p.pet\_id

AND p.type\_id=pt.type\_id AND t.appointment\_id=a.appointment\_id

GROUP BY t.branch\_id, pt.type\_name

ORDER BY t.branch\_id, SUM(t.total\_amount) DESC;

**Sample output :**

****

**View 2: The purpose of this view is to display the amount of appointments made during morning times of each branch in last year**

CREATE OR REPLACE VIEW morningApp AS

SELECT t.branch\_id, COUNT(t.transaction\_id) AS MORNING

FROM appointment a, transaction t

WHERE t.appointment\_id = a.appointment\_id AND

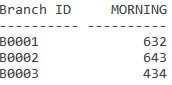
EXTRACT(HOUR FROM CAST(a.appointment\_datetime AS TIMESTAMP)) BETWEEN 10 AND 12

AND EXTRACT(YEAR FROM a.appointment\_datetime) = EXTRACT(YEAR FROM SYSDATE)-1

GROUP BY t.branch\_id

ORDER BY t.branch\_id;

**Sample output :**

****

**View 3: The purpose of this view is to display the amount of appointments made during afternoon times of each branch in last year**

CREATE OR REPLACE VIEW afternoonApp AS

SELECT t.branch\_id, COUNT(t.transaction\_id) AS AFTERNOON

FROM appointment a, transaction t

WHERE t.appointment\_id = a.appointment\_id AND

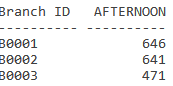
EXTRACT(HOUR FROM CAST(a.appointment\_datetime AS TIMESTAMP)) BETWEEN 13 AND 15

AND EXTRACT(YEAR FROM a.appointment\_datetime) = EXTRACT(YEAR FROM SYSDATE)-1

GROUP BY t.branch\_id

ORDER BY t.branch\_id;

**Sample output :**

****

**View 4: The purpose of this view is to display the amount of appointments made during evening times of each branch in last year**

CREATE OR REPLACE VIEW eveningApp AS

SELECT t.branch\_id, COUNT(t.transaction\_id) AS EVENING

FROM appointment a, transaction t

WHERE t.appointment\_id = a.appointment\_id AND

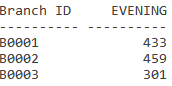
EXTRACT(HOUR FROM CAST(a.appointment\_datetime AS TIMESTAMP)) BETWEEN 16 AND 18

AND EXTRACT(YEAR FROM a.appointment\_datetime) = EXTRACT(YEAR FROM SYSDATE)-1

GROUP BY t.branch\_id

ORDER BY t.branch\_id;

**Sample output :**

****

**View 5: This purpose of this view is to calculate the sales of each branch in the year 2020 first half**

CREATE OR REPLACE VIEW Sales2020\_1stHalf AS

SELECT branch\_id, SUM(total\_amount) AS Sales2020\_1stHalf

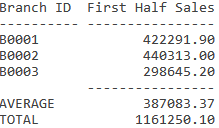
FROM transaction

WHERE EXTRACT(YEAR FROM transaction\_dateTime) = 2020 AND EXTRACT(MONTH FROM transaction\_dateTime) <= 6

GROUP BY branch\_id

ORDER BY branch\_id, SUM(total\_amount) DESC;

**Sample output :**



**View 6: This purpose of this view is to calculate the sales of each branch in the year 2020 second half**

CREATE OR REPLACE VIEW Sales2020\_2ndHalf AS

SELECT branch\_id, SUM(total\_amount) AS Sales2020\_2ndHalf

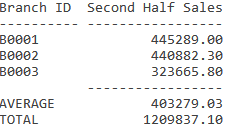
FROM transaction

WHERE EXTRACT(YEAR FROM transaction\_dateTime) = 2020 AND EXTRACT(MONTH FROM transaction\_dateTime) > 6

GROUP BY branch\_id

ORDER BY branch\_id, SUM(total\_amount) DESC;

**Sample output :**

****

**5.1.2 User Defined Exceptions**

**Exception 1: This exception is defined in the procedure add appointment and it will be raised when treatment ID enter by user is not found or invalid**

e\_invalid\_treatment EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_treatment, -20050);

RAISE\_APPLICATION\_ERROR(-20050, 'Invalid Treatment ID.');

**Exception 2: This exception is defined in the procedure add appointment and it will be raised when pet ID enter by user is not found or invalid**

e\_invalid\_pet EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_pet, -20051);

RAISE\_APPLICATION\_ERROR(-20051, 'Invalid Pet ID.');

**Exception 3: This exception is defined in the summary, detail, and on demand report. It will be raised when the record of report generate by user was not found**

e\_norecord EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_norecord,-20060);

RAISE\_APPLICATION\_ERROR(-20060,'No record found');

**Exception 4: This exception is defined in trigger appointment date time and it will be raised when the appointment insert is before now.**

RAISE\_APPLICATION\_ERROR(-20052, 'Cannot insert the date time before now.' );

**Exception 5: This exception is defined in trigger appointment date time and it will be raised when the appointment insert is not within business hours.**

RAISE\_APPLICATION\_ERROR(-20053, 'Date time must be within business hour.' );

**Exception 6: This exception is defined in trigger delete appointment and it will be raised when the appointment is recorded in transaction and unable to delete**

RAISE\_APPLICATION\_ERROR(-20055,'Appointment delete unsuccessful');

**5.1.3 Sequences**

**Sequence 1: This sequence will automatically generate the appointment ID for the Kuala Lumpur branch. It will be used when inserting a new appointment in the KL branch and adding 1 after the next new appointment.**

CREATE SEQUENCE app\_seq\_KL

MINVALUE 10000001

MAXVALUE 99999999

START WITH 10000001

INCREMENT BY 1;

**Sequence 2: This sequence will automatically generate the appointment ID for the Pulau Pinang branch. It will be used when inserting a new appointment in the PG branch and adding 1 after the next new appointment.**

CREATE SEQUENCE app\_seq\_PG

MINVALUE 10000001

MAXVALUE 99999999

START WITH 10000001

INCREMENT BY 1;

**Sequence 3: This sequence will automatically generate the appointment ID for the Kedah branch. It will be used when inserting a new appointment in the KD branch and adding 1 after the next new appointment.**

CREATE SEQUENCE app\_seq\_KD

MINVALUE 10000001

MAXVALUE 99999999

START WITH 10000001

INCREMENT BY 1;

**5.1.4 Triggers**

**Trigger 1: This trigger is use to validate the age of the veterinarian that should be above 22 years old when a new veterinarian is inserted into the database**

CREATE OR REPLACE TRIGGER trgVetAge

BEFORE INSERT OR UPDATE ON Veterinarian

FOR EACH ROW

BEGIN

IF((ROUND((SYSDATE-:new.vet\_dob)/365)) < 22) THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Veterinarian must be at least 22 years old.' );

END IF;

END;

/

**5.2 (Tan Teoh Xin Ee)**

**5.2.1 Views**

**View 1: The purpose of this view is to store the information of medicine details from different branches. For example, the medicine name, qty and amount.**

create or replace view medicalUsed As

select t.branch\_id, d.medic\_id, m.medic\_name, sum(d.line\_qty) as quantity, sum(d.line\_total) as amount

from transaction t, branch b, transactiondetail d, medicalsupply m

where b.branch\_id = t.branch\_id

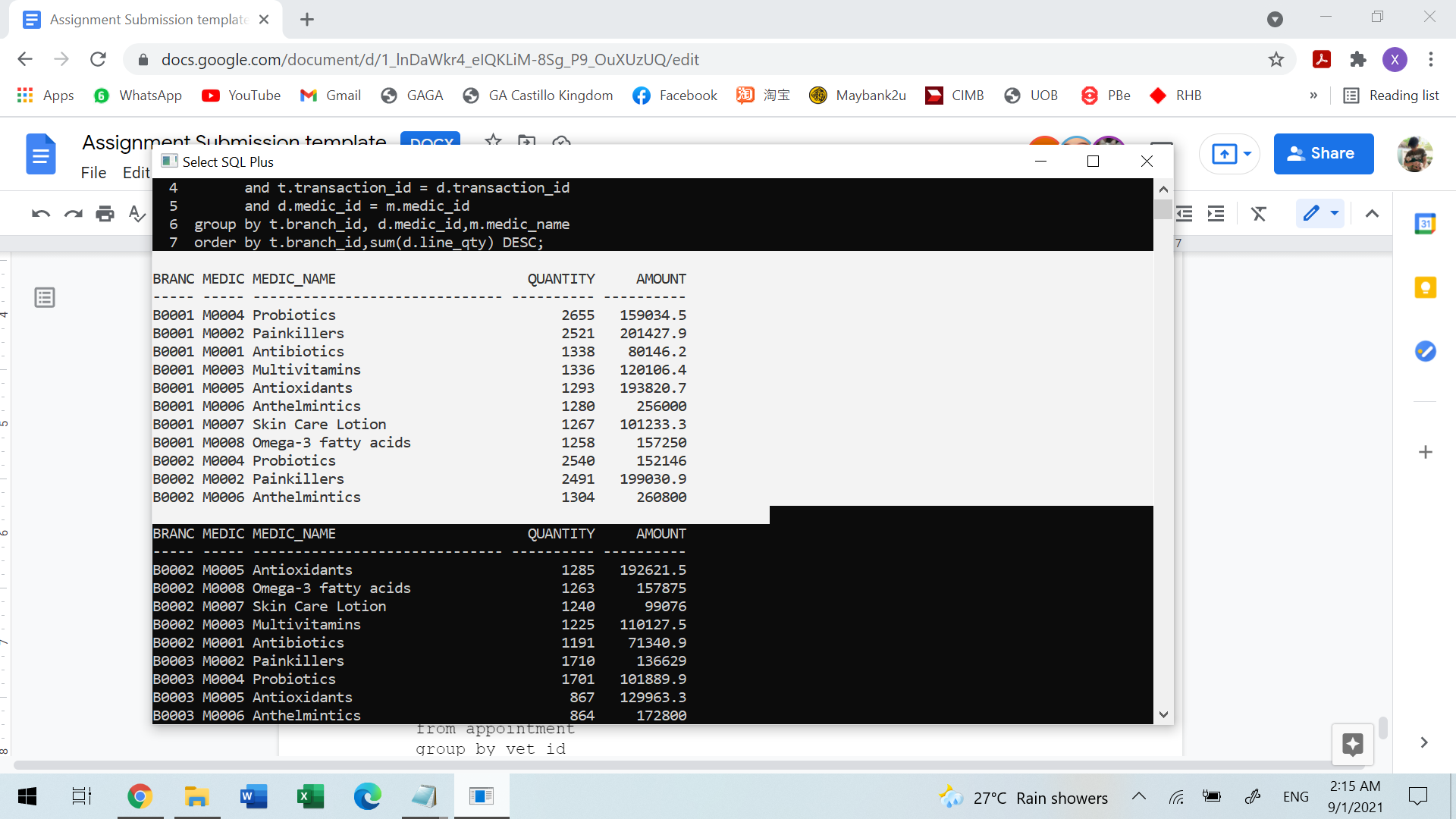
and t.transaction\_id = d.transaction\_id

and d.medic\_id = m.medic\_id

group by t.branch\_id, d.medic\_id,m.medic\_name

order by t.branch\_id,sum(d.line\_qty) DESC;

**Sample Output:**

****

**View 2: The purpose of this view is to store the numbers of appointments received by every veterinarian.**

create or replace view appointNum As

select count(appointment\_id) as NoOfapp, vet\_id

from appointment

group by vet\_id

order by count(appointment\_id) desc;

**Sample Output:**

****

**View 3: The purpose of this view is to store late sent stock information from the supplier such as purchase date, receive date and the duration from purchase date until receive date. The stock normally will be sent within 6 days.**

create or replace view difdate as

select purchase\_id, supplier\_id, purchase\_date, receive\_date,(receive\_date-purchase\_date) as duration

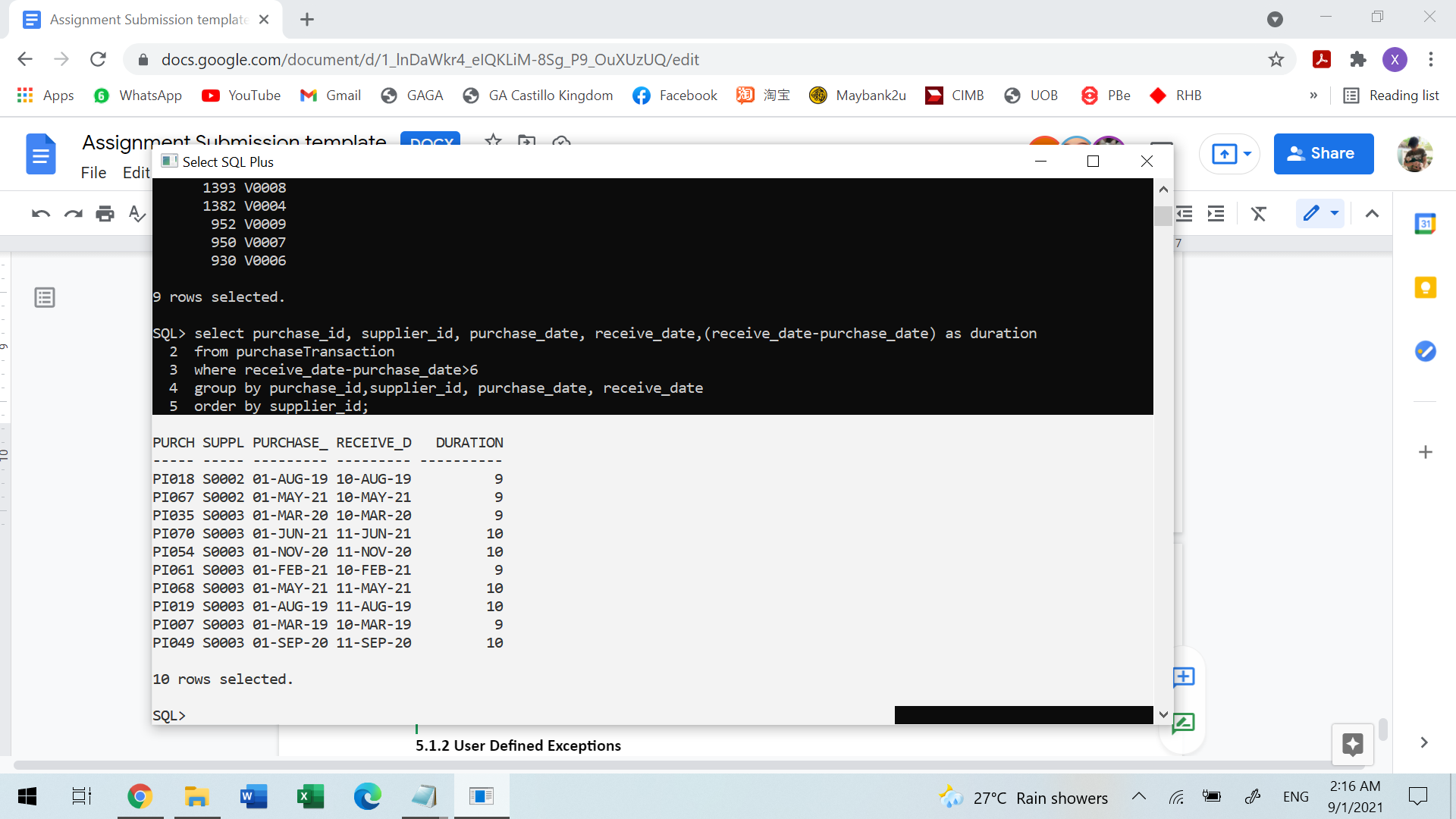
from purchaseTransaction

where receive\_date-purchase\_date>6

group by purchase\_id,supplier\_id, purchase\_date, receive\_date

order by supplier\_id;

**Sample Output:**

****

**5.2.2 User Defined Exceptions**

**The purpose of this exception is to prompt the user that ‘Invalid supplier code.’, if he/she key in the wrong supplier id.**

EXCE\_SUPPLIERCODE EXCEPTION;

PRAGMA EXCEPTION\_INIT(EXCE\_SUPPLIERCODE, -20310);

**5.2.3 Sequence**

**The purpose of this sequence is to generate numbers for medic\_id, which will be needed in the medic add procedure.**

CREATE SEQUENCE MEDICID

MINVALUE 8

MAXVALUE 9999

START WITH 8

INCREMENT BY 1;

**5.2.4 Trigger**

**The purpose of this trigger is to delete the amount of purchase items from the amount of purchase transaction, if the purchase item was deleted.**

CREATE OR REPLACE TRIGGER trgDelPurchaseItem

After Delete ON PurchaseItem

FOR EACH ROW

BEGIN

Update PurchaseTransaction

SET purchase\_amount = purchase\_amount - (:new.purchase\_qty \* :new.purchase\_price)

where purchase\_id = :new.purchase\_id;

END;

/

**5.3 (Tan Wei Siong)**

**5.3.1 User Defined Exceptions**

**Exception 1: This exception is defined in the trigger check appointment date time to check whether the selected time for the veterinarian is book or not. If the time has been booked, the exception will raise and output the suggested time for the user.**

Date\_Time\_Booked EXCEPTION;

PRAGMA exception\_init(Date\_Time\_Booked, -20200 );

**Exception 2: This exception is defined in the procedure pet register to check whether the provided owner information exists or not. It will raise when the owner information is not found.**

No\_owner\_found EXCEPTION;

PRAGMA exception\_init(No\_owner\_found,-20201);

**Exception 3: This exception is defined in the on-demand report of the pet treatment detail. The exception will raise when the user enters the invalid pet id into the system. It will message the user that the pet is not found.**

NO\_PET\_FOUND EXCEPTION;

PRAGMA EXCEPTION\_INIT(NO\_PET\_FOUND, -20202);

**Exception 4: This exception is defined in the trigger check owner age. The exception will raise when the registered user age is below 18.**

RAISE\_APPLICATION\_ERROR(-20004, 'Pet Owner must be at least 18 years old.' );

**5.3.2 Sequence**

**Sequence 1: This sequence is used to auto generate the pet owner id. The sequence number will increase 1 when there is a new pet owner registered.**

CREATE SEQUENCE owner\_seq

START WITH 501

INCREMENT BY 1;

**Sequence 2: This sequence is used to auto generate the pet id. The sequence number will increase 1 when there is a new pet registered.**

CREATE SEQUENCE pet\_seq

START WITH 1001

INCREMENT BY 1;

**5.3.3 Procedure**

**5.3.3.1 Procedure 1: Pet Owner Registration**

**Purpose: The purpose of this procedure is to let the staff register the pet owner in an easier way. The staff online need to input the owner information into this procedure and the owner will be added into the database.**

**Procedure code:**

CREATE OR REPLACE Procedure Prc\_register\_owner(in\_owner\_Name IN VARCHAR2, in\_owner\_Contact IN VARCHAR2,

in\_owner\_dob IN Date, in\_gender IN CHAR, in\_state IN VARCHAR2,

in\_city IN VARCHAR2, in\_postcode IN VARCHAR2,

in\_streetName IN VARCHAR2) AS

v\_owner\_seq VARCHAR2(4);

v\_petOwner\_id PetOwner.owner\_id%TYPE;

BEGIN

SELECT to\_char(lpad(owner\_seq.nextval,'4','0')) INTO v\_owner\_seq from dual;

v\_petOwner\_id := ('O'||v\_owner\_seq);

Insert into petOwner values(v\_petOwner\_id, in\_owner\_Name , in\_owner\_Contact, in\_owner\_dob,

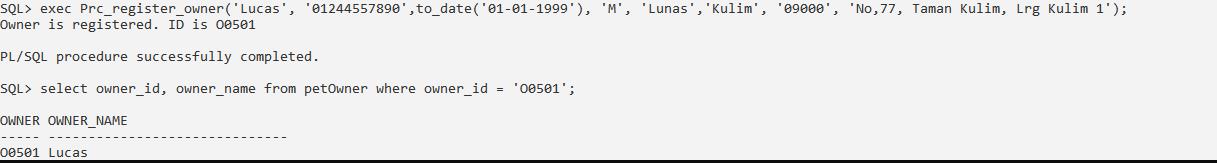
in\_gender, in\_state, in\_city, in\_postcode, in\_streetName);

dbms\_output.put\_line('Owner is registered. ID is ' || v\_petOwner\_id);

END;

/

**Sample Output:**



**5.3.4 Trigger**

**Trigger 1: The purpose of this trigger is to auto update the purchase amount when there is a new purchase item inserted into the purchase item table.**

CREATE OR REPLACE TRIGGER trgPurchaseItem

After Insert ON PurchaseItem

FOR EACH ROW

BEGIN

Update PurchaseTransaction

SET purchase\_amount = purchase\_amount + (:new.purchase\_qty \* :new.purchase\_price)

where purchase\_id = :new.purchase\_id;

END;

/

**5.4 (Nigel Lee Jian Hsee)**

**5.4.1 Views**

**View 1: The purpose of this view is to store the treatment revenue in the year of 2021 of each branch.**

CREATE OR REPLACE VIEW FullRevenueGroupByTreatment2021 AS

select T.branch\_id,TT.treatment\_id, TT.treatment\_type, sum(T.total\_amount) as Total\_Amount

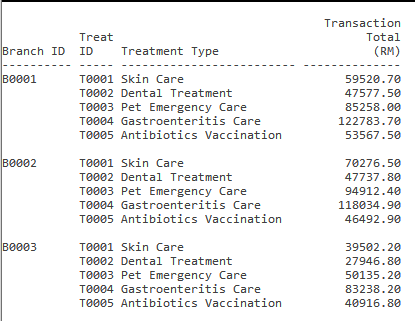
from transaction T, appointment A, treatment TT

where A.appointment\_id = T.appointment\_id AND A.treatment\_id = TT.treatment\_id AND EXTRACT(YEAR FROM T.transaction\_datetime) = '2021'

group by T.branch\_id,TT.treatment\_id, TT.treatment\_type

order by branch\_id;

**Sample Output:**

****

**View 1: The purpose of this view is to store the medicine revenue of each treatment in the year of 2021 of each branch. For example, total medicine revenue that earned after a dental treatment.**

CREATE OR REPLACE VIEW MedicRevenueGroupByTreatment2021 AS

select T.branch\_id,TT.treatment\_id, TT.treatment\_type, sum(TD.line\_total) AS Medic\_Revenue, sum(TD.line\_qty) AS Sold\_Quantity, (sum(TD.line\_total)/sum(TD.line\_qty)) AS Revenue\_Per\_Quantity

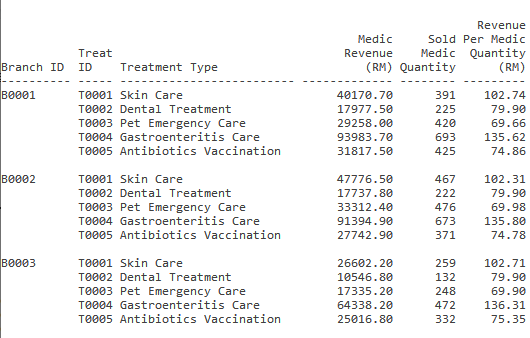
from transactiondetail TD, transaction T, appointment A, treatment TT

where TD.transaction\_id = T.transaction\_id AND A.appointment\_id = T.appointment\_id AND A.treatment\_id = TT.treatment\_id AND EXTRACT(YEAR FROM T.transaction\_datetime) = '2021'

group by T.branch\_id,TT.treatment\_id, TT.treatment\_type

order by branch\_id;

**Sample Output:**

****

**5.4.2 User Defined Exceptions**

**Exception 1: This exception is defined in the procedure add and create transaction to check whether the entered appointment exists. If not an error message will be displayed.**

e\_invalid\_appointmentid EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_appointmentid, -20150);

**Exception 2: This exception is defined in the procedure add and create transaction to check whether the entered appointment id transaction is already created . If yes an error message will be displayed.**

e\_repeated\_transaction EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_repeated\_transaction, -20151);

**Exception 3: This exception is defined in the adding transaction detail procedure. The exception will raise when the user adds a medicine that already exists in a created transaction. If yes an error message will be prompted.**

e\_repeated\_medicid EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_repeated\_medicid, -20155);

**Exception 4: This exception is defined in the adding transaction detail procedure. The exception will raise when the medic quantity entered is less than or equal zero.**

e\_zero\_qty EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_zero\_qty, -20156);

**Exception 5: This exception is defined in the edit transaction detail procedure. The exception will raise when the transaction is an old transaction that has already passed 7 days.**

e\_dayexceed EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_dayexceed, -20157)

**Exception 5: This exception is defined in the edit transaction detail procedure. The exception will raise when the user enters a transaction detail that does not exist.**

e\_invalid\_transactiondetail EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_transactiondetail, -20158);

**Exception 6: This exception is defined in the edit transaction detail procedure. The exception will raise when the quantity is less than zero.**

e\_lesszero\_qty EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_lesszero\_qty, -20159);

**Exception 7: This exception is defined in the edit transaction detail procedure. The exception will raise when the medicine quantity entered is the same as the old quantity and remind the user that the quantity is the same and will not change.**

e\_samequantity EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_samequantity, -20161);

**Exception 8: This exception is defined in the vet monthly performance summary report. The exception will raise when the entered vet id is invalid.**

e\_invalid\_vetid EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_vetid, -20162);

**Exception 9: This exception is defined in the vet monthly performance summary report. The exception will raise when the entered month is invalid.**

e\_invalid\_month EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_month, -20163);

**Exception 10: This exception is defined in the branch performance yearly detail report and in the vet monthly performance summary report. The exception will raise when the year is invalid.**

e\_invalid\_year EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_year, -20164);

**Exception 11: This exception is defined in the branch poor performance on demand report. The exception will raise when the entered branch id is invalid.**

e\_invalid\_branch EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_branch, -20165);

**Exception 12: This exception is defined in the branch performance yearly detail report. The exception will raise when the entered year is inavlid and out of range.**

e\_invalid\_year EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_year, -20166);

**5.4.3 Trigger**

**Trigger 1: The purpose of this trigger is to validate pet age by checking the date of birth. Which means when the date of birth is after the system, an error will be raised to prompt the user. The pet age must be more than 1.**

CREATE OR REPLACE TRIGGER trgPetAge

BEFORE INSERT OR UPDATE ON Pet

FOR EACH ROW

BEGIN

IF((ROUND((SYSDATE-:new.pet\_dob)/365)) < 0) THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Pet must be at least more than 0 years old.' );

END IF;

END;

/