Table of Content

Chapter 1: The Case Study

1.1 Background of the system

Bus Ticketing System is one of the systems which provide efficiency management and help the bus company to handle daily operation easily and efficiently. It help to enhanced company's daily routine scheduling. By using bus ticketing system, performing tasks has never been easier, it provide facility to selling tickets on each counter or over the internet. Besides, administrator staff can be maintain all the data by using oracle database such as maintain staff details, main bus details, maintain route details, maintain bus schedule, and so on. Hundreds thousands of data in the database can be retrieve faster and display in different level, different output to help bus company in better decision making result in improve company's profit.

The bus ticketing system that we do is based on a new express route bus company which started operation at the end of year 2012. This company is a new startup so there only few routes services provided over a domestic privatized transportation the west Malaysia. In our scenario, there will have 2 years records stored in the database in order for analysis and decision making.

1.2 Class Diagram

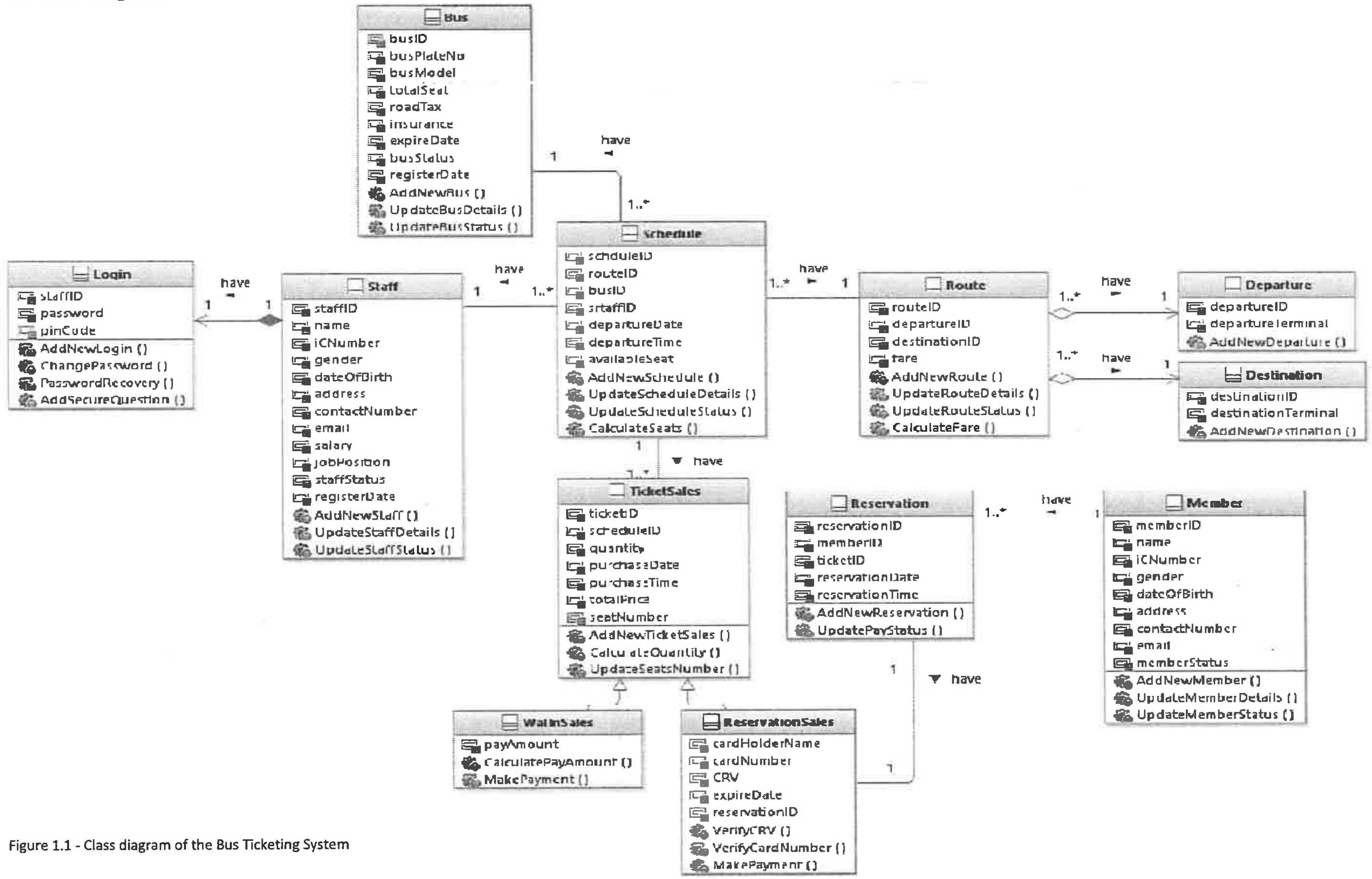
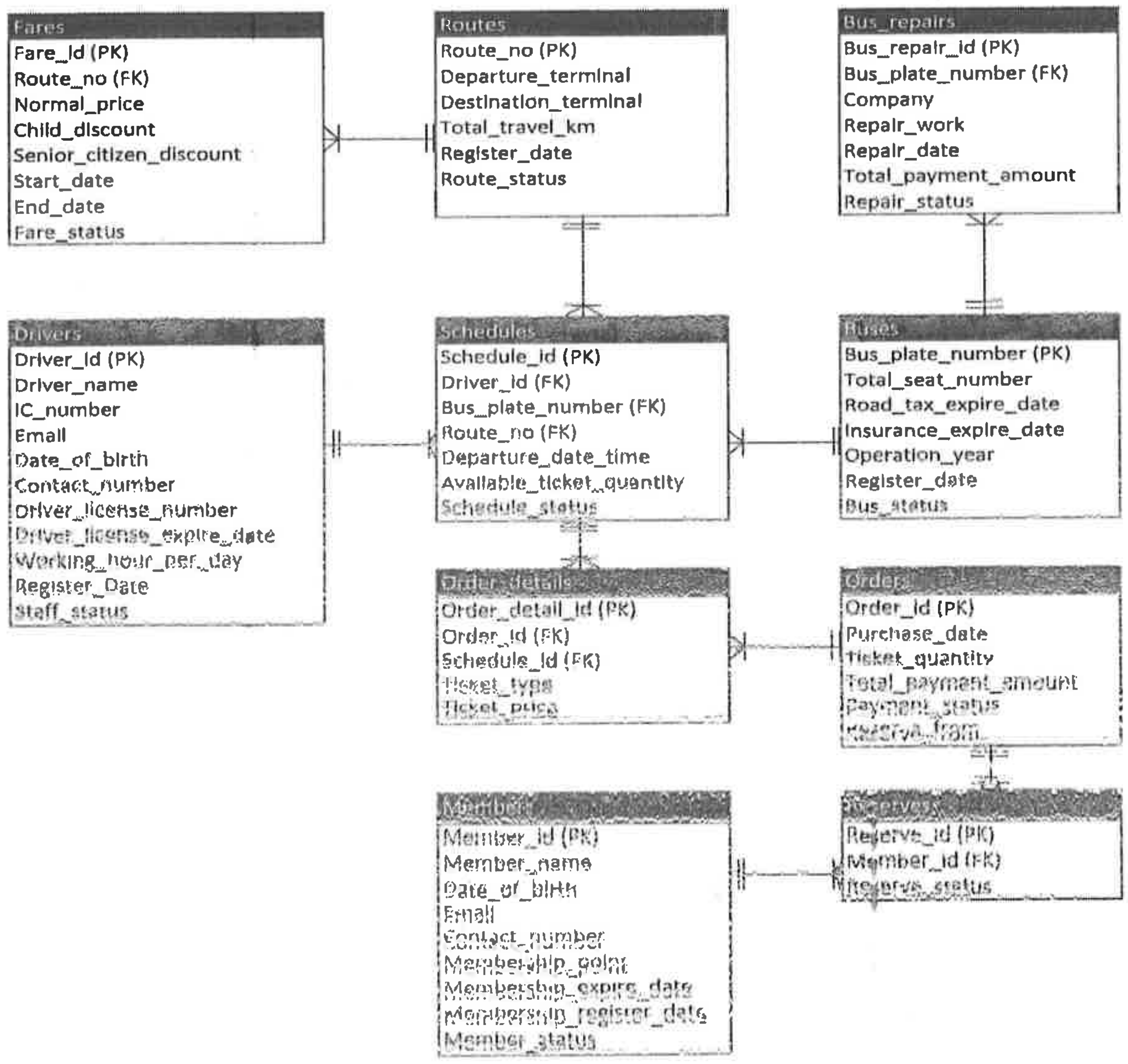


Figure 1.1 - Class diagram of the Bus Ticketing System

Chapter 2: Enhancements to the system

2.1 The improved system ERD



2.2 Assumptions

1. ROUTES is important for Bus Company to provide transportation to the customer. Thus ROUTE table is created to store the route that will provided by this company to the customer. Each ROUTES had provided 2 way from different departure terminal and destination terminal. So there will be One to many relationship from ROUTES table to SCHEDULES table and One schedule only been assigned one ROUTES at a time.

2. Since high level management may require to analysis the fare based on different session to make decision on increasing or decrease the fare of each route. So FARES table is used to keep track the change of each time the fare of a specific route is changed. That's mean one ROUTES can have many fare change history records in the Fares table but one FARES records is only keep track one of the Route.

3. Driver table is used to store the driver's information and also have a one to many relationship with the Schedule table because one Driver can only in charge one Schedule at same time but the driver can be assigned many different time schedules.

4. BUSES is a transport that provide service from one departure terminal to the destination terminal. BUSES table is create for store it information such road tax expire date, insurance expire date and on.

5. Every Bus has need to send for check every year or when found out some problem with the bus. Thurs BUS_REPAIRS table is created for keep track the repair detail of each time the bus send for repair. Each BUSES can have many repair record in the BUS_REPAIRS.

6. For a safe journal, normally will assign the same driver and bus on the same route in a SCHEDULES table because the drive is well know about the route start from departure terminal to destination terminal. Based on the experience that the staff handle this route, the journal will be more safety to the customers.

7. To improve efficiency, ORDERS table is use to store both online purchase and counter purchase, column reserve_from in the ORDERS table was use to indicate whether it was online or counter purchase. If column reserve_from store as NULL, it refer to counter purchase while reserve_from store reserve_id from table RESERVES refer to online purchase. Column reserve_from is not a foreign key of column reserev_id in table RESERVES in order to store NULL value.

8. ORDER_DETAILS table is child of ORDERS table which used to indicate the ticket type, schedule id, ticket price of each ORDER_DETAILS row.

9. One ORDER_DETAILS row refer to one ticket, if one ORDER_ID have 5 ORDER_DETAIL_ID it mean that the order was 5 tickets. Each ORDER_DETAILS refer to one schedules, one single ticket type, and one single ticket price.

10. Column TOTAL_PAYMENT_AMOUNT in ORDERS is use to store the total ticket price from table ORDER_DETAILS.

11. Counter purchase in table ORDER must pay first so the payment_status will be 'Paid' while online purchase can pay within a weeks or two so payment_status of online purchase can be either 'Paid' or 'Unpaid'.

12. Customer can cancel their online purchase and the reserve_status in table RESERVES will stated as 'Cancelled', while payment is waiting, reserve_status will stated as 'Pending' and 'Completed' as payment completed.

2.3 Business Rules

1. Each member can online book up to 10 tickets at a time.
2. Online purchase can delay their payment within a weeks or two.
3. Online purchase will auto cancelled if payment does not receive within set period of time.
4. Counter purchase must made payment on the spot.
5. Staff must over 18 years old in order to register as a staff in this bus company.
6. To register as member must be more than 18 years old.
7. Bus seat number cannot more than 100.
8. Maximum of membership point is 999999 points.
9. Bus fare can be 0 if the company promote with free seat.
10. A bus with expired on road tax, insurance or repairing is not allow to assign to the schedules.
11. A driver with resigned or inactive status is not allow to assign to the schedules.
12. Available seat cannot more than the maximum bus seat.
13. Company will gift free membership point for those member who often make ticket reservation with paid status.
14. Member can exchange their 100 membership point for 20% discount.
15. Ticket price will differetent depends on ticket type such Normal, Child and Senior Citizen.
16. Customer will get a discount from the normal price if it is a children or senior citizen.
17. Staff working hours must between 1 to 12 hours.
18. Member's membership was expired is not allow to make online purchase.
19. Each completed online purchase will awarded 10 membership point.
20. Bus and driver cannot have a duplicate time schedule at the same time.
21. Bus operation year exceed 15 years, road tax expired or insurance expired will change status to "Unavailable".
22. Driver license was expired, all his/her schedule will become "Unavailable".
23. Schedule that past will change it status to "Unavailable" or available seat is equal zero will change it status to "Full" and it's not available for user make order on this schedule.

2.4 Database Designation Language (DBDL)

```
CREATE TABLE drivers(
driver_id                     VARCHAR(5)    NOT NULL,
driver_name                   VARCHAR(50)   NOT NULL,
ic_number                     VARCHAR(14)   NOT NULL      UNIQUE,
email                         VARCHAR(30)   NOT NULL      UNIQUE,      (21 records)
date_of_birth                 DATE          NOT NULL,
contact_number                VARCHAR(12)   NOT NULL      UNIQUE,
driver_license_number         VARCHAR(10)   NOT NULL      UNIQUE,
driver_license_expire_date    DATE          NOT NULL,
working_hour_per_day          NUMBER(2)     NOT NULL      CHECK (working_hour_per_day
BETWEEN 1 AND 12),
register_date                 DATE          NOT NULL,
staff_status                  VARCHAR(25)   NOT NULL,
PRIMARY KEY (driver_id)
);

CREATE TABLE buses(                                              (20 records)
bus_plate_number              VARCHAR(8)    NOT NULL,
total_seat_number             NUMBER(3)     NOT NULL      CHECK(total_seat_number
BETWEEN 40 AND 100),
road_tax_expire_date          DATE          NOT NULL,
insurance_expire_date         DATE          NOT NULL,
operation_year                NUMBER(3)     NOT NULL      CHECK(operation_year>=0),
register_date                 DATE          NOT NULL,
bus_status                    VARCHAR(20)   NOT NULL,
PRIMARY KEY (bus_plate_number)
);

CREATE TABLE bus_repairs(
bus_repair_id                 VARCHAR(6)    NOT NULL,        (76 records)
bus_plate_number              VARCHAR(8)    NOT NULL,
company_name                  VARCHAR(50)   NOT NULL,
repair_work                   VARCHAR(50)   NOT NULL,
repair_date                   DATE          NOT NULL,
total_payment_amount          NUMBER(7,2)   NOT NULL      CHECK(total_payment_amount >=
0),
repair_status                 VARCHAR(50)   NOT NULL,
PRIMARY KEY (bus_repair_id),
FOREIGN KEY(bus_plate_number) REFERENCES buses(bus_plate_number)
);

CREATE TABLE members(
member_id                     VARCHAR(14)   NOT NULL,        (21 records)
member_name                   VARCHAR(50)   NOT NULL,
date_of_birth                 DATE          NOT NULL,
email                         VARCHAR(30)   NOT NULL      UNIQUE,
contact_number                VARCHAR(12)   NOT NULL,
membership_point              NUMBER(6)     NOT NULL      CHECK(membership_point
BETWEEN 0 AND 999999),
membership_expire_date        DATE          NOT NULL,
membership_register_date      DATE          NOT NULL,
member_status                 VARCHAR(20)   NOT NULL,
PRIMARY KEY(member_id)
);

CREATE TABLE routes(
route_no                      VARCHAR(3)    NOT NULL,
departure_terminal            VARCHAR(20)   NOT NULL,       (6 records)
destination_terminal          VARCHAR(20)   NOT NULL,
total_travel_km               NUMBER(4,1)   NOT NULL        CHECK (total_travel_km > 0),
register_date                 DATE          NOT NULL,
route_status                  VARCHAR(10)   NOT NULL,
CONSTRAINT uc_route UNIQUE (departure_terminal,destination_terminal),
PRIMARY KEY (route_no)
);

CREATE TABLE fares(
fare_id                       VARCHAR(5)    NOT NULL,       (21 records)
route_no                      VARCHAR(3)    NOT NULL,
normal_price                  NUMBER(5,2)   NOT NULL        CHECK (normal_price >= 0),
child_discount                NUMBER(5,2)   DEFAULT 30      CHECK (child_discount >= 0),
senior_citizen_discount       NUMBER(5,2)   DEFAULT 50      CHECK (senior_citizen_discount >=
0),
start_date                    DATE          NOT NULL,
end_date                      DATE          DEFAULT NULL,
fare_status                   VARCHAR(7)    DEFAULT 'Current',
PRIMARY KEY (fare_id),
FOREIGN KEY(route_no) REFERENCES routes(route_no)
);

CREATE TABLE schedules(
schedule_id                   VARCHAR(10)   NOT NULL,
driver_id                     VARCHAR(5)    NOT NULL,       (2184 records)
bus_plate_number              VARCHAR(8)    NOT NULL,
route_no                      VARCHAR(3)    NOT NULL,
departure_date_time           DATE          NOT NULL,
available_ticket_quantity     number(2)     NOT NULL        CHECK (available_ticket_quantity
BETWEEN 0 AND 100),
schedule_status               VARCHAR(20)   NOT NULL,
PRIMARY KEY (schedule_id),
FOREIGN KEY(driver_id) REFERENCES drivers(driver_id),
FOREIGN KEY(bus_plate_number) REFERENCES buses(bus_plate_number),
FOREIGN KEY(route_no) REFERENCES routes(route_no)
);

CREATE TABLE reserves(                                       (826 records)
reserve_id                    VARCHAR(10)   NOT NULL,
member_id                     VARCHAR(14)   NOT NULL,
reserve_status                VARCHAR(10)   DEFAULT 'Pending',
PRIMARY KEY (reserve_id),
FOREIGN KEY (member_id) REFERENCES members(member_id)
);

CREATE TABLE orders(
order_id                VARCHAR(10)   NOT NULL,         (40988 records)
purchase_date           DATE          NOT NULL,
ticket_quantity         NUMBER(2)     NOT NULL        CHECK (ticket_quantity BETWEEN 1
AND 10),
total_payment_amount    NUMBER(6,2)   NOT NULL        CHECK (total_payment_amount >=
0),
payment_status          VARCHAR(6)    NOT NULL,
reserve_from            VARCHAR(5)    DEFAULT NULL,
PRIMARY KEY (order_id)
);

CREATE TABLE order_details(
order_detail_id         VARCHAR(10)   NOT NULL,
order_id                VARCHAR(10)   NOT NULL,         (81016 records)
schedule_id             VARCHAR(5)    NOT NULL,
ticket_type             VARCHAR(15)   NOT NULL,
ticket_price            NUMBER(5,2)   NOT NULL,       CHECK(ticket_price >= 0),
PRIMARY KEY (order_detail_id),
FOREIGN KEY (order_id) REFERENCES orders(order_id),
FOREIGN KEY (schedule_id) REFERENCES schedules(schedule_id)
);
```

Chapter 3: Queries, Procedures, Triggers and Reports

3.1 (Chong Kar Ming)

3.1.1 Query 1: 2 Years Route's Sales Comparison

Purpose: The purpose of this query is to display the all routes total sales comparison between 2 years input by the user.

SQL statement:

```
SELECT r.route_no, r.departure_terminal,
       r.destination_terminal,
       fy.total_amount as First_Year_Sales_Amount,
       sy.total_amount as Second_Year_Sales_Amount
FROM   routes r, YearSales fy, YearSales sy
WHERE (fy.route_no = r.route_no) AND
      (fy.year = &FirstYear) AND
      (sy.route_no = r.route_no) AND
      (sy.year = &SecondYear);
```

Sample Output:

```
SQL> @e:\query1.txt
Enter value for firstyear: 2013
old   4:         (fy.year = &FirstYear) AND
new   4:         (fy.year = 2013) AND
Enter value for secondyear: 2014
old   6:         (sy.year = &SecondYear)
new   6:         (sy.year = 2014)

ROU DEPARTURE_TERMINAL   DESTINATION_TERMINAL FIRST_YEAR_ SECOND_YEAR
--- -------------------- -------------------- ----------- -----------
R12 Kuala Lumpur         Bandar Melaka         119,463.56  109,718.01
R15 Bandar Melaka        Kuala Lumpur          121,736.36  110,305.10
R14 Georgetown           Kuala Lumpur          119,056.93  109,265.06
R11 Kuala Lumpur         Georgetown            119,130.37  111,020.20
R13 Johor Bharu          Kuala Lumpur          120,869.23  110,417.19
R10 Kuala Lumpur         Johor Bharu           120,475.91  109,428.52

6 rows selected.

SQL> spool off
```

3.1.2 Query 2: Monthly Routes Sales

Purpose: The purpose of this query is to display the monthly total sales from highest to lowest based on the specified route and year that input by the user.

SQL statement:

```
SELECT    TO_CHAR(TO_DATE(EXTRACT(MONTH FROM s.departure_date_time),'mm'),'MON') AS Month,
          r.route_no, r.departure_terminal, r.destination_terminal,
          COUNT(od.order_detail_id) AS Total_Ticket_Sold,
          TO_CHAR(SUM(od.ticket_price),'999,999.99') AS Total_Sales_Amount
FROM      routes r, schedules s, order_details od, orders o
WHERE    (s.route_no = '&RouteNo') AND
         (od.schedule_id = s.schedule_id) AND
         (o.order_id = od.order_id) AND
         (o.payment_status = 'Paid') AND
         (EXTRACT(YEAR FROM s.departure_date_time) = &Year) AND
         (r.route_no = s.route_no)
GROUP BY  EXTRACT(MONTH FROM s.departure_date_time), r.route_no, r.departure_terminal,
          r.destination_terminal
ORDER BY  6 DESC;
```

Sample Output:

```
SQL> @e:\query2.txt
Enter value for routeno: R13
old   4: WHERE (s.route_no = '&RouteNo') AND
new   4: WHERE (s.route_no = 'R13') AND
Enter value for year: 2014
old   8:        (EXTRACT(YEAR FROM s.departure_date_time) = &Year) AND
new   8:        (EXTRACT(YEAR FROM s.departure_date_time) = 2014) AND

MONTH        ROU DEPARTURE_TERMINAL   DESTINATION_TERMINAL TOTAL_TICKET_SOLD TOTAL_SALES
------------ --- -------------------- -------------------- ----------------- -----------
JUN          R13 Johor Bharu          Kuala Lumpur                       576   11,010.67
FEB          R13 Johor Bharu          Kuala Lumpur                       577   10,856.75
NOV          R13 Johor Bharu          Kuala Lumpur                       575   10,803.67
OCT          R13 Johor Bharu          Kuala Lumpur                       572   10,586.71
APR          R13 Johor Bharu          Kuala Lumpur                       518    9,902.75
JAN          R13 Johor Bharu          Kuala Lumpur                       509    9,706.23
AUG          R13 Johor Bharu          Kuala Lumpur                       502    9,640.16
JUL          R13 Johor Bharu          Kuala Lumpur                       505    9,584.14
MAR          R13 Johor Bharu          Kuala Lumpur                       510    9,550.44
SEP          R13 Johor Bharu          Kuala Lumpur                       505    9,489.41
MAY          R13 Johor Bharu          Kuala Lumpur                       490    9,286.26

11 rows selected.

SQL> spool off
```

3.1.3 Query 3: Recent Route Fares Changes

Purpose: The purpose of this query is to display the number of recent route fares changes records based on the specified route and required number records that input by the user.

SQL statement:

```
SELECT *
FROM(
   SELECT    f.fare_id, f.start_date, f.end_date, r.route_no,
             r.departure_terminal, r.destination_terminal,
             TO_CHAR(f.normal_price,'999.99') as Normal_Price,
             TO_CHAR(f.normal_price * ((100-f.child_discount) / 100),'999.99') as Child_Price,
             TO_CHAR(f.normal_price * ((100-f.senior_citizen_discount) / 100),'999.99') as
             Senior_Citizen_Price, f.fare_status
   FROM      fares f, routes r
   WHERE    (f.route_no = '&RouteNo') AND
            (r.route_no = f.route_no)
   ORDER BY 2 DESC
)
WHERE        rownum <= &RecentRecord
ORDER BY     2 DESC;
```

Sample Output:

```
SQL> @e:\query3.txt
Enter value for routeno: R14
old   9:    WHERE (f.route_no = '&RouteNo') AND
new   9:    WHERE (f.route_no = 'R14') AND
Enter value for recentrecords: 4
old  13: WHERE rownum <= &RecentRecords
new  13: WHERE rownum <= 4

FARE_ START_DATE END_DATE   ROU DEPARTURE_TERMINAL   DESTINATION_TERMINAL NORMAL_ CHILD_P SENIOR_ FARE_ST
----- ---------- ---------- --- -------------------- -------------------- ------- ------- ------- -------
F1021 01-06-2014            R14 Georgetown           Kuala Lumpur           30.00   18.00   12.00 Current
F1017 01-01-2014 31-05-2014 R14 Georgetown           Kuala Lumpur           32.00   19.20   12.80 Past
F1012 01-01-2013 31-12-2013 R14 Georgetown           Kuala Lumpur           19.00   13.30    9.50 Past
F1005 01-10-2012 31-12-2012 R14 Georgetown           Kuala Lumpur           24.00   16.80   12.00 Past

SQL> spool off
```

3.1.4 Procedure 1: Insert New Route and Fares

Purpose: The purpose of this procedure is to insert a new route and come together with a new fares records. User is require input the departure terminal, destination terminal, distance and fare rate when call this procedure. If the route does not exist in the database, this procedure will generate new unique primary key for both new route and fare records.

SQL statement:

```
CREATE OR REPLACE PROCEDURE proc_InsertRoute (In_Departure IN VARCHAR,
                                              In_Destination IN VARCHAR,
                                              In_Distance IN VARCHAR,
                                              In_Rate IN VARCHAR) AS

v_newRouteNo            routes.route_no%TYPE;
v_newFareId             fares.fare_id%TYPE;
v_normalPrice           fares.normal_price%TYPE;
e_invalidDistanceInput  exception;
e_invalidDistanceLength exception;
e_invalidRateInput      exception;

BEGIN
  IF (is_number(In_Distance) = false) THEN
      raise e_invalidDistanceInput;
  ELSE
     IF (In_Distance <= 0) THEN
         raise e_invalidDistanceLength;
     ELSE
        IF (is_number(In_Rate) = false) THEN
            raise e_invalidRateInput;
        END IF;
      END IF;
  END IF;

  v_newRouteNo  := 'R'||route_no.nextval;
  v_newFareId   := 'F'||fare_id.nextval;
  v_normalPrice := ROUND(TO_NUMBER(In_Distance) * TO_NUMBER(In_Rate),2);
  Insert into routes values(v_newRouteNo, in_departure, In_Destination,
                            In_Distance, to_char(sysdate,'dd-mm-yyyy'), 'Active');
  Insert into fares values(v_newFareId, v_newRouteNo, v_normalPrice, default,
                           default, to_char(sysdate,'dd-mm-yyyy'), null, 'Current');

  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('New route added successfully.');
  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('Route No: '||v_newRouteNo);
  DBMS_OUTPUT.PUT_LINE('Departure Terminal: '||In_departure);
  DBMS_OUTPUT.PUT_LINE('Destination Terminal: '||In_destination);
  DBMS_OUTPUT.PUT_LINE('Distance in KM: '||In_Distance);
  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('New fare added successfully for this new route.');
  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('Fare ID: '||v_newFareId);
  DBMS_OUTPUT.PUT_LINE('Normal Price: '||v_normalPrice);
  DBMS_OUTPUT.PUT_LINE('Child Discount Rate: 30%');
  DBMS_OUTPUT.PUT_LINE('Senior Citizen Discount Rate: 50%');

EXCEPTION
   WHEN e_invalidDistanceInput THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Input: Input distance must be a numeric.');
   WHEN e_invalidDistanceLength THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Input: Input distance must greater than 0.');
   WHEN e_invalidRateInput THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Input: Input rate must be a numeric.');
   WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Record insert failed: This route already exist.');
        rollback;
END;
/
```

Sample Output:

```
SQL> exec proc_insertRoute('Kuala Lumpur','Perak','254','0.1')


New route added successfully.


Route No: R16
Departure Terminal: Kuala Lumpur
Destination Terminal: Perak
Distance in KM: 254


New fare added successfully for this new route.


Fare ID: F1022
Normal Price: 25.4
Child Discount Rate: 30%
Senior Citizen Discount Rate: 50%

PL/SQL procedure successfully completed.

SQL> spool off
```

3.1.5 Procedure 2: Update All Fares by Percentage

Purpose: The purpose of this procedure is update all the route fares based on the percentage input by the user. If user input a positive percentage number then it will increase all the route fares else if user input a negative percentage number then it will decrease all the route fares.

SQL statement:

```
CREATE OR REPLACE PROCEDURE proc_UpdateFare(In_Percent IN VARCHAR) AS

v_newFareID     fares.fare_id%TYPE;
v_oldPrice      fares.normal_price%TYPE;
v_newPrice      fares.normal_price%TYPE;
v_fareNo        varchar(4);
v_newNumber     number;
v_recordCount   number;
e_invalidInput  exception;

CURSOR fare_cursor IS
    SELECT   f.fare_id, f.route_no,f.normal_price, f.child_discount, f.senior_citizen_discount,
             r.departure_terminal, r.destination_terminal
    FROM     fares f, routes r
    WHERE    (r.route_no = f.route_no) AND (f.fare_status = 'Current')
    ORDER BY 2,1;

    fare_rec fare_cursor%ROWTYPE;

BEGIN
  IF (is_number(In_Percent) = false) THEN
      raise e_invalidInput;
  END IF;

  v_recordCount := 0;

  DBMS_OUTPUT.PUT_LINE(chr(10));
```

```
IF(In_Percent >=0) THEN
   DBMS_OUTPUT.PUT_LINE('Increased percent of price: '||In_Percent||'%');
ELSE
   DBMS_OUTPUT.PUT_LINE('Decreased percent of price: '||In_Percent||'%');
END IF;

DBMS_OUTPUT.PUT_LINE(chr(10));
DBMS_OUTPUT.PUT_LINE('Fare ID '||'Route No '||'Departure Terminal '||'Destination Terminal '||
                     'Old Price '||'New Price ');
DBMS_OUTPUT.PUT_LINE('======= '||'======== '||'================== '||'==================== '||
                     '========= '||'========= ');

OPEN fare_cursor;
LOOP
FETCH fare_cursor INTO fare_rec;
EXIT WHEN fare_cursor%NOTFOUND;
   v_oldPrice   := fare_rec.normal_price;
   v_newPrice   := (v_oldPrice * ((100 + TO_NUMBER(In_Percent))/100));
   v_newFareID := 'F'||fare_id.nextval;

   UPDATE fares
   SET    fare_status = 'Past', end_date = TO_CHAR(sysdate,'dd-mm-yyyy')
   WHERE  fare_id = fare_rec.fare_id;

   INSERT INTO fares VALUES(v_newFareID,fare_rec.route_no, v_newPrice,fare_rec.child_discount,
                            fare_rec.senior_citizen_discount, TO_CHAR(sysdate,'dd-mm-yyyy'),
                            null,'Current');

   v_newNumber := v_newNumber + 1;

   DBMS_OUTPUT.PUT_LINE(RPAD(v_newFareID,8,' ')||RPAD(fare_rec.route_no,9,' ')||
                        RPAD(fare_rec.departure_terminal,19,' ')||
                        RPAD(fare_rec.destination_terminal,21,' ')||'RM'||
                        LPAD(TO_CHAR(v_oldPrice,'999.99'),7,' ')||' RM'||
                        LPAD(TO_CHAR(v_newPrice,'999.99'),7,' '));
```

```
  v_recordCount := v_recordCount + 1;

  END LOOP;

  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('Records updated: '||v_recordCount);

EXCEPTION
   WHEN e_invalidInput THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Input: Input percentage must be a numeric.');
   WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error Found. Please contact your Database Administrator');

END;
/
```

Sample Output:

```
SQL> exec proc_updateFare(10)


Increased percent of price: 10%


Fare ID Route No Departure Terminal Destination Terminal Old Price New Price
======= ======== ================== ==================== ========= =========
F1023   R10      Kuala Lumpur       Johor Bharu          RM  10.00 RM  11.00
F1024   R11      Kuala Lumpur       Georgetown           RM  20.00 RM  22.00
F1025   R12      Kuala Lumpur       Bandar Melaka        RM  26.00 RM  28.60
F1026   R13      Johor Bharu        Kuala Lumpur         RM  20.00 RM  22.00
F1027   R14      Georgetown         Kuala Lumpur         RM  30.00 RM  33.00
F1028   R15      Bandar Melaka      Kuala Lumpur         RM  23.00 RM  25.30
F1029   R16      Kuala Lumpur       Perak                RM  25.40 RM  27.94


Records updated: 7

PL/SQL procedure successfully completed.

SQL> spool off
```

3.1.6 Trigger 1: Validate Insert Order Detail

Purpose: The purpose of this trigger is to validate whether the schedule is available for purchase while user insert an order detail record. If the schedule status is "Full" or past with status "Unavailable", it will stop user insert the record into database and display a warning message tell the user why unable to insert the records. If all the condition is valid, it will automatically generate a new unique primary key for this order detail record.

Trigger code:

```
CREATE OR REPLACE TRIGGER trg_validateInsert
BEFORE INSERT ON order_details
FOR EACH ROW
DECLARE
  v_scheduleStatus       schedules.schedule_status%TYPE;
  v_normalPrice          fares.normal_price%TYPE;
  v_childPrice           fares.child_discount%TYPE;
  v_seniorCenPrice       fares.senior_citizen_discount%TYPE;
  v_orderDetailID        order_details.order_detail_id%TYPE;
  e_scheduleUnavailable  exception;
BEGIN
  SELECT schedule_status
  INTO   v_scheduleStatus
  FROM   schedules
  WHERE (schedule_id = :new.schedule_id);

  IF SQL%FOUND THEN
     IF NOT (v_scheduleStatus = 'Available') THEN
        raise e_scheduleUnavailable;
     ELSE
        SELECT f.normal_price, ((normal_Price * (100 - f.child_discount)) / 100),
               ((normal_Price * (100 - f.senior_citizen_discount)) / 100)
        INTO   v_normalPrice, v_childPrice, v_seniorCenPrice
        FROM   schedules s, fares f
        WHERE (s.schedule_id = :new.schedule_id) AND
               (f.route_no = s.route_no) AND (f.fare_status = 'Current');
        IF (:new.ticket_type = 'Normal') THEN
            :new.ticket_price := v_normalPrice;
        ELSE
           IF(:new.ticket_type = 'Child') THEN
              :new.ticket_price := v_childPrice;
           ELSE
              :new.ticket_price := v_seniorCenPrice;
           END IF;
        END IF;
        v_orderDetailID := 'OT'||order_detail_id.nextval;
        :new.order_detail_id := v_orderDetailID;
     END IF;
  END IF;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
        raise_application_error( -20001, 'Records cannot be insert: Schedule is not available.');
   WHEN e_scheduleUnavailable THEN
        raise_application_error( -20001, 'Records cannot be insert: Schedule is '||
                                'v_scheduleStatus);
   WHEN OTHERS THEN
       raise_application_error( -20001, 'Error Found. Please contact your Database
                                Administrator');
END;
/
```

Sample Output:

```
SQL> Insert into order_details values('OT82017','O41981','S3002','Child',0.0);
Insert into order_details values('OT82017','O41981','S3002','Child',0.0)
            *
ERROR at line 1:
ORA-20001: Records cannot be insert: Schedule is Unavailable
ORA-06512: at "ADRIEL.TRG_VALIDATEINSERT", line 42
ORA-04088: error during execution of trigger 'ADRIEL.TRG_VALIDATEINSERT'


SQL> spool off
```

Sample Output:

```
SQL> Insert into order_details values('OT82017','O41981','S3102','Child',0.0);
Insert into order_details values('OT82017','O41981','S3102','Child',0.0)
*
ERROR at line 1:
ORA-20001: Records cannot be insert: Schedule is Full
ORA-06512: at "ADRIEL.TRG_VALIDATEINSERT", line 42
ORA-04088: error during execution of trigger 'ADRIEL.TRG_VALIDATEINSERT'


SQL> spool off
```

3.1.7 Trigger 2: Update Seat Availability and Order Total Payment

Purpose: The purpose of this trigger is to update the seat availability of the schedule listed in the new inserted order detail record. Besides, it will also update a new total payment amount on the particular order record. If user change the ticket type, it will automatically retrieve the fare based on the type changed by the user and update the total payment amount on the particular order record.

Trigger code:

```
CREATE OR REPLACE TRIGGER trg_updateSeat
AFTER INSERT OR UPDATE OF ticket_type ON order_details
FOR EACH ROW
DECLARE
  v_availableTicketQuantity schedules.available_ticket_quantity%TYPE;
  v_normalPrice             fares.normal_price%TYPE;
  v_childDiscount           fares.child_discount%TYPE;
  v_seniorCenDiscount       fares.senior_citizen_discount%TYPE;
  v_price                   order_details.ticket_price%TYPE;

BEGIN

     CASE
         WHEN INSERTING THEN

              UPDATE schedules
              SET    available_ticket_quantity = available_ticket_quantity - 1
              WHERE  schedule_id = :new.schedule_id;

              UPDATE orders
              SET    ticket_quantity = ticket_quantity + 1,
                     total_payment_amount = total_payment_amount + :new.ticket_price
              WHERE  order_id = :new.order_id;

              SELECT available_ticket_quantity INTO v_availableTicketQuantity
              FROM   schedules
              WHERE  schedule_id = :new.schedule_id;
```

```
            IF (v_availableTicketQuantity = 0) THEN
                UPDATE schedules
                SET    schedule_status = 'Full'
                WHERE  schedule_id = :new.schedule_id;
            END IF;

        WHEN UPDATING THEN

            SELECT f.normal_price, f.child_discount, f.senior_citizen_discount
            INTO   v_normalPrice, v_childDiscount, v_seniorCenDiscount
            FROM   schedules s, fares f
            WHERE  (s.schedule_id = :new.schedule_id) AND
                   (f.route_no = s.route_no) AND (f.fare_status = 'Current');

            IF (:new.ticket_type = 'Normal') THEN
                v_price := v_normalPrice;
            ELSE
               IF(:new.ticket_type = 'Child') THEN
                   v_price := ((v_normalPrice * (100 - v_childDiscount)) / 100);
               ELSE
                   v_price := ((v_normalPrice * (100 - v_seniorCenDiscount)) / 100);
               END IF;
            END IF;

            UPDATE orders    .
            SET    total_payment_amount = total_payment_amount - :old.ticket_price + v_price
            WHERE  order_id = :new.order_id;
      END CASE;

EXCEPTION
   WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error Found. Please contact your Database Administrator');
END;
/
```

3.1.8 Report 1: Detail Report of Route Fares Details

Purpose: The purpose of this report is to display the route fares details based on the specified route and year range input by the user. By using this report, user can know how long ago about the change of the last fares and also check the previous fares change history to make decision about the fares changing of this particular route.

PL/SQL code:

```
CREATE OR REPLACE PROCEDURE proc_detailReport(In_FirstYear IN VARCHAR, In_SecondYear IN VARCHAR,
                                               In_RouteNo IN VARCHAR) AS

v_departureTerminal    routes.departure_terminal%TYPE;
v_destinationTerminal  routes.destination_Terminal%TYPE;
v_date                 fares.end_date%TYPE;
v_fareID               fares.fare_id%TYPE;
v_currentDate          fares.start_date%TYPE;
v_latestDate           fares.start_date%TYPE;
v_different            NUMBER;
v_recordCount          NUMBER := 0;
v_loop                 NUMBER := 0;
v_pageCount            NUMBER := 1;
v_totalSales           NUMBER(8,2);
v_statement            VARCHAR(10);
e_invalidInput         exception;
e_invalidYear          exception;

CURSOR fare_cursor IS

   SELECT fare_id, start_date, end_date, TO_CHAR(normal_price,'99.99') as Normal_price, fare_status,
          child_discount,
          TO_CHAR((normal_price*(100-child_discount)/100),'99.99') AS Child_Price,
          senior_citizen_discount,
          TO_CHAR((normal_price*(100-senior_citizen_discount)/100),'99.99') AS Senior_Citizen_Price
   FROM   fares
   WHERE  (route_no = In_RouteNo) AND
          (EXTRACT(YEAR FROM start_date) BETWEEN In_FirstYear AND In_SecondYear);
   fare_rec fare_cursor%ROWTYPE;

BEGIN

   IF ((is_number(In_FirstYear) = false) OR (is_number(In_SecondYear) = false)) THEN
        raise e_invalidInput;
   ELSE
      IF ((In_FirstYear > TO_CHAR(sysdate,'yyyy')) OR (In_SecondYear > TO_CHAR(sysdate,'yyyy'))) THEN
           raise e_invalidYear;
      END IF;
   END IF;

  v_statement:= 'Route';

  SELECT departure_terminal, destination_terminal
  INTO   v_departureTerminal, v_destinationTerminal
  FROM   routes
  WHERE  route_no = In_RouteNo;

  v_statement:= 'Fares';

  SELECT fare_id, start_date into v_fareID, v_latestDate
  FROM   fares
  WHERE (route_no = In_RouteNo) AND (fare_status = 'Current');

  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('        '||LPAD(' ',37,'  ')||'Route Fare Details Report');
  DBMS_OUTPUT.PUT_LINE('        '||LPAD(' ',37,'  ')||LPAD('=',25,'='));

  OPEN  fare_cursor;
  LOOP
  FETCH fare_cursor INTO fare_rec;
  EXIT  WHEN fare_cursor%NOTFOUND;
```

```
IF (v_loop = 0 or v_loop = 20) THEN
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE('Printed Date: '||TO_CHAR(sysdate,'dd-mm-yyyy')||LPAD(' ',74,' ')||
                         'Page: '||LPAD(v_pageCount,2,' '));
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE('Route No: '||In_RouteNo);
    DBMS_OUTPUT.PUT_LINE('Departure Terminal: '|| v_departureTerminal);
    DBMS_OUTPUT.PUT_LINE('Destination Terminal: '|| v_destinationTerminal);
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE('Fare ID '||'Start Date '||'End Date    '||'Normal Price '||
                         'Child Price '||'Senior Citizen Price '||
                         'Total Sales Amount '||'Fare Status');
    DBMS_OUTPUT.PUT_LINE('======= '||'========== '||'========== '||'============ '||
                         '=========== '||'==================== '||'================== '||
                         '===========');
    v_loop := 0;
    v_pageCount := v_pageCount + 1;

END IF;

IF (fare_rec.end_date IS null) THEN
    v_date := to_char(sysdate,'dd-mm-yyyy');
ELSE
    v_date := fare_rec.end_date;
END IF;

v_statement:= 'Sales';
SELECT  SUM(od.ticket_price) AS Total_Sales_Amount
INTO    v_totalSales
FROM    schedules s, order_details od, orders o
WHERE   (s.route_no = In_RouteNo) AND
        (od.schedule_id = s.schedule_id) AND
        (o.order_id = od.order_id) AND
        (o.payment_status = 'Paid') AND
        (TO_CHAR(s.departure_date_time,'dd-mm-yyyy') BETWEEN fare_rec.start_date AND v_date);
    IF (v_totalSales IS NULL) THEN
        v_totalSales := 0.0;
    END IF;

    DBMS_OUTPUT.PUT_LINE(RPAD(fare_rec.fare_id,8,' ')||
                         RPAD(fare_rec.start_date,11,' ')||
                         RPAD(v_date,11,' ')||
                         LPAD('RM',4,' ')||LPAD(fare_rec.normal_price,8,' ')||
                         LPAD('RM',4,' ')||LPAD(fare_rec.child_price,8,' ')||
                         LPAD('RM',13,' ')||LPAD(fare_rec.senior_citizen_price,8,' ')||
                         LPAD('RM',7,' ')||LPAD(TO_CHAR(v_totalSales,'999,999.99'),12,' ')||' '||
                         RPAD(fare_rec.fare_status,12,' '));

    v_loop := v_loop + 1;
    v_recordCount := v_recordCount + 1;

  END LOOP;

  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('Records found: '||v_recordCount);
  DBMS_OUTPUT.PUT_LINE(chr(10));

  v_different := trunc(to_date(sysdate,'dd-mm-yyyy') - v_latestDate);

  IF (v_different < 30) THEN
      DBMS_OUTPUT.PUT_LINE('Latest fare update: '||v_different||' day(s) ago');
  ELSE
     IF (v_different < 356) THEN
         v_different := ROUND(MONTHS_BETWEEN(sysdate, v_latestDate),0);
         DBMS_OUTPUT.PUT_LINE('Latest fare update: '||v_different||' month(s) ago');
     ELSE
         v_different := ROUND((MONTHS_BETWEEN(sysdate, v_latestDate)/12),0);
         DBMS_OUTPUT.PUT_LINE('Latest fare update: '||v_different||' year(s) ago');
     END IF;
  END IF;
  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('        '||LPAD(' ',42,' ')||'-End of report-');

EXCEPTION
   WHEN e_invalidInput THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Input: Input year must be a numeric.');
   WHEN e_invalidYear THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Input: Input year cannot be future year.');
   WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Report produce failed: No '|| v_statement || ' records found in
                             database.');
   WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error Found. Please contact your Database Administrator');
END;
/
```

Sample Output:

```
SQL> exec proc_detailReport('2012','2014','R14')

                                                  Route Fare Details Report
                                                  =========================


Printed Date: 27-11-2014                                                                                  Page:  1


Route No: R14
Departure Terminal: Georgetown
Destination Terminal: Kuala Lumpur


Fare ID Start Date End Date    Normal Price Child Frice Senior Citizen Price Total Sales Amount Fare Status
======= ========== ========== ============ =========== ==================== ================== ===========
F1005   01-10-2012 31-12-2012   RM   24.00 RM    16.80           RM    12.00     RM   21,933.00 Past
F1012   01-01-2013 31-12-2013   RM   19.00 RM    13.30           RM     9.50     RM  119,056.93 Past
F1017   01-01-2014 31-05-2014   RM   32.00 RM    19.20           RM    12.80     RM   48,075.89 Past
F1021   01-06-2014 27-11-2014   RM   30.00 RM    18.00           RM    12.00     RM   61,189.17 Current


Records found: 4


Latest fare update: 6 month(s) ago


                                                       -End of report-

PL/SQL procedure successfully completed.

SQL> spool off
```

3.1.9 Report 2: On Demand Report of Monthly Route Sales

Purpose: The purpose of this query is display monthly route total sales based on the specified route and year input by the user. By using this report, user can know which month or season have a highest or lowest sales compare with other month within the year. User can make a decision for increase the fares or schedule for the high sales months for gain more profit.

PL/SQL code:

```
CREATE OR REPLACE PROCEDURE proc_onDemandReport(In_Year IN VARCHAR, In_RouteNo IN VARCHAR) AS

v_departureTerminal     routes.departure_terminal%TYPE;
v_destinationTerminal   routes.destination_terminal%TYPE;
v_AvailableTicket       NUMBER;
v_pageCount             NUMBER := 1;
v_monthCount            NUMBER := 0;
v_highestSales          NUMBER := 0;
v_lowestSales           NUMBER := 0;
v_ticketLeft            NUMBER := 0;
v_ticketSold            NUMBER := 0;
v_salesAmount           NUMBER(9,2) := 0;
v_highestMonth          VARCHAR(5);
v_lowestMonth           VARCHAR(5);
v_statement             VARCHAR(10);
e_invalidInput          exception;
e_invalidYear           exception;

CURSOR schedule_cursor IS
       SELECT   EXTRACT(month from s.departure_date_time) as Month,
                COUNT(od.order_detail_id) as Total_Ticket_Sold,
                SUM(od.ticket_price) as Total_Sales_Amount
       FROM     schedules s, order_details od. orders o
       WHERE   (s.schedule_id = od.schedule_id) AND
               (s.route_no = In_RouteNo) AND
               (o.order_id = od.order_id) AND
               (o.payment_status = 'Paid') AND
               (EXTRACT(year from s.departure_date_time) = In_Year)
       GROUP BY EXTRACT(month from s.departure_date_time)
       ORDER BY 1;

       schedule_rec schedule_cursor%ROWTYPE;
BEGIN

  IF (is_number(In_Year) = false) THEN
      raise e_invalidInput;
  ELSE
     IF (In_Year > TO_CHAR(sysdate,'yyyy')) THEN
         raise e_invalidYear;
     END IF;
  END IF;

  v_statement := 'Route';

  SELECT departure_terminal, destination_terminal
  INTO   v_departureTerminal, v_destinationTerminal
  FROM   routes
  WHERE  route_no = In_RouteNo;

  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('        '||'Route Year '| In_Year||' Monthly Sales Details Report');
  DBMS_OUTPUT.PUT_LINE('        '||LPAD('=',44,'='));
  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('Printed Date: '||TO_CHAR(sysdate,'dd-mm-yyyy')||
                       LPAD(' ',31,' ')||'Page: ' |LPAD(v_pageCount,2,' '));
  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('Route No: '||In_RouteNo);
  DBMS_OUTPUT.PUT_LINE('Departure Terminal: '|| v_departureTerminal);
  DBMS_OUTPUT.PUT_LINE('Destination Terminal: '|| v_destinationTerminal);
  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('Month    '||'Total Ticket Left '||
                       'Total Tickets Sold '||'Total Sales Amount');
  DBMS_OUTPUT.PUT_LINE('======= '||'================= '||'================== '||'==================');
```

```
OPEN schedule_cursor;
LOOP
FETCH schedule_cursor INTO schedule_rec;
EXIT WHEN schedule_cursor%NOTFOUND;
   SELECT SUM(available_ticket_quantity)
   INTO   v_AvailableTicket
   FROM   schedules
   WHERE  (route_no = In_RouteNo) AND
          (EXTRACT(year from departure_date_time) = In_Year) AND
          (EXTRACT(month from departure_date_time) = schedule_rec.Month);

   DBMS_OUTPUT.PUT_LINE(RPAD(TO_CHAR(TO_DATE(schedule_rec.Month,'MM'),'MON'),8,' ')||
                        LPAD(TO_CHAR(v_AvailableTicket,'999,999'),17,' ')||
                        LPAD(TO_CHAR(schedule_rec.Total_Ticket_Sold,'999,999'),19,' ')||
                        LPAD('RM',8,'')||
                        LPAD(TO_CHAR(schedule_rec.Total_Sales_Amount,'999,999.99'),11,' '));

   v_salesAmount  := v_salesAmount + schedule_rec.Total_Sales_Amount;
   v_ticketLeft   := v_ticketLeft + v_availableTicket;
   v_ticketSold   := v_ticketSold + schedule_rec.Total_Ticket_Sold;
   v_monthCount   := v_monthCount + 1;

   IF (schedule_rec.Total_Sales_Amount > v_highestSales) THEN
       v_highestSales := schedule_rec.Total_Sales_Amount;
       v_highestMonth := TO_CHAR(TO_DATE(schedule_rec.Month,'MM'),'MON');
   END IF;

   IF (v_lowestSales = 0) THEN
       v_lowestSales := schedule_rec.Total_Sales_Amount;
       v_lowestMonth := TO_CHAR(TO_DATE(schedule_rec.Month,'MM'),'MON');
   ELSE
      IF(schedule_rec.Total_Sales_Amount < v_lowestSales) THEN
         v_lowestSales  := schedule_rec.Total_Sales_Amount;
         v_lowestMonth  := TO_CHAR(TO_DATE(schedule_rec.Month,'MM'),'MON');
      END IF;
   END IF;
```

```
  END LOOP;
  CLOSE schedule_cursor;

  DBMS_OUTPUT.PUT_LINE('         '||'----------------- '||'------------------ '||'------------------');
  DBMS_OUTPUT.PUT_LINE('         '||LPAD(TO_CHAR(v_ticketLeft,'999,999'),17,' ')||
                       LPAD(TO_CHAR(v_ticketSold,'999,999'),19,' ')||
                       LPAD('RM',8,' ')||LPAD(TO_CHAR(v_salesAmount,'999,999.99'),11,' '));
  DBMS_OUTPUT.PUT_LINE('         '||'----------------- '||'------------------ '||'------------------');
  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('Highest Sales Amount  : RM'||TO_CHAR(v_highestSales,'9,999,999.99')||'       '||
                       'Month : '||v_highestMonth);
  DBMS_OUTPUT.PUT_LINE('Lowest Sales Amount   : RM'||TO_CHAR(v_lowestSales,'9,999,999.99')||'       '||
                       'Month : '||v_lowestMonth);
  DBMS_OUTPUT.PUT_LINE('Avarage Sales Amount  : RM'||
                       TO_CHAR(v_salesAmount/v_monthCount,'9,999,999.99'));
  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('         '||LPAD(' ',14,' ')||'-End of report-');

EXCEPTION
   WHEN e_invalidInput THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Input: Input year must be a numeric.');
   WHEN e_invalidYear THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Input: Input year cannot be future year.');
   WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Report produce failed: No '|| v_statement ||
                             ' records found in database.');
   WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error Found. Please contact your Database Administrator');

END;
/
```

Sample Output:

```
SQL> exec proc_onDemandReport('2014','R13')


         Route Year 2014 Monthly Sales Details Report
         ============================================


Printed Date: 27-11-2014                                Page:  1


Route No: R13
Departure Terminal: Johor Bharu
Destination Terminal: Kuala Lumpur


Month   Total Ticket Left Total Tickets Sold Total Sales Amount
======= ================= ================== ==================
JAN                    93                509      RM   9,706.23
FEB                    25                577      RM  10,856.75
MAR                    92                510      RM   9,550.44
APR                    84                518      RM   9,902.75
MAY                   112                490      RM   9,286.26
JUN                    26                576      RM  11,010.67
JUL                    97                505      RM   9,584.14
AUG                   100                502      RM   9,640.16
SEP                    97                505      RM   9,489.41
OCT                    31                572      RM  10,586.71
NOV                    28                575      RM  10,803.67
        ----------------- ------------------ ------------------
                      785              5,839      RM 110,417.19
        ----------------- ------------------ ------------------


Highest Sales Amount   : RM    11,010.67      Month : JUN
Lowest Sales Amount    : RM     9,286.26      Month : MAY
Avarage Sales Amount   : RM    10,037.93


                    -End of report-

PL/SQL procedure successfully completed.

SQL> spool off
```

3.1.10 Report 3: Summary report of Yearly Route Total Sales

Purpose: The purpose of this query is to display all routes total sales compare between the 2 years that input by the user. It have display the percentage change by each of the routes total sales between 2 years. By using this report, user can easily look out which route is profit or loss based on comparison total sales between 2 years and make decision to take action on higher sales or lower sales route such increase more schedule on the higher sales route or make a serious decision on the lower sales route is the gap is too large between 2 years.

PL/SQL code:

```
CREATE OR REPLACE PROCEDURE proc_summaryReport(In_FirstYear IN VARCHAR, In_SecondYear IN VARCHAR) AS

v_status                  VARCHAR(10);
v_pageCount               NUMBER := 1;
v_loop                    NUMBER := 0;
v_firstYearAmount         NUMBER(9,2);
v_secondYearAmount        NUMBER(9,2);
v_firstYearTotalAmount    NUMBER(9,2)  := 0;
v_SecondYearTotalAmount   NUMBER(9,2)  := 0;
v_yearChange              NUMBER(5,2)  := 0;
v_totalYearChange         NUMBER(5,2)  := 0;
e_invalidInput            exception;
e_invalidYear             exception;

CURSOR route_cursor IS
       SELECT route_no, departure_terminal, destination_terminal
       FROM   routes;

       route_rec route_cursor%ROWTYPE;

BEGIN
   IF ((is_number(In_FirstYear) = false) OR (is_number(In_SecondYear) = false)) THEN
        raise e_invalidInput;
   ELSE
      IF ((In_FirstYear > TO_CHAR(sysdate,'yyyy')) OR
          (In_SecondYear > TO_CHAR(sysdate,'yyyy'))) THEN
           raise e_invalidYear;
      END IF;
   END IF;

  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('     '||LPAD(' ',20,' ')||'Route Yearly Sales Summary Report');
  DBMS_OUTPUT.PUT_LINE('     '||LPAD(' ',20,' ')||LPAD('=',33,'='));

  OPEN  route_cursor;
  LOOP
  FETCH route_cursor INTO route_rec;
  EXIT  WHEN route_cursor%NOTFOUND;

     IF (v_loop = 0 or v_loop = 20) THEN

         DBMS_OUTPUT.PUT_LINE(chr(10));
         DBMS_OUTPUT.PUT_LINE(RPAD('Printed Date: ',14,' ')||
                              RPAD(TO_CHAR(sysdate,'dd-mm-yyyy'),10,' ')||
                              LPAD(' ',57,' ')||'Page: '||LPAD(v_pageCount,2,' '));
         DBMS_OUTPUT.PUT_LINE(chr(10));
         DBMS_OUTPUT.PUT_LINE('Route No '||'Departure Terminal '||'Destination Terminal '||
                              'Year '||In_FirstYear||' Sales '||
                              'Year '||In_SecondYear||' Sales '||'Changes ');
         DBMS_OUTPUT.PUT_LINE('======== '||'================== '||
                               '==================== '||'=============== '||
                               '=============== '||'========');
         v_loop := 0;
         v_pageCount := v_pageCount + 1;
      END IF;

      SELECT  SUM(od.ticket_price) as Total_Amount into v_firstYearAmount
      FROM    schedules s, order_details od, orders o
      WHERE  (s.schedule_id = od.schedule_id) AND
             (o.order_id = od.order_id) AND
             (o.payment_status = 'Paid') AND
             (s.route_no = route_rec.route_no) AND
             (EXTRACT(YEAR FROM s.departure_date_time) = In_FirstYear);
```

```
SELECT  SUM(od.ticket_price) as Total_Amcunt into v_secondYearAmount
FROM    schedules s, order_details od, orders o
WHERE   (s.schedule_id = od.schedule_id) AND
        (o.order_id = od.order_id) AND
        (o.payment_status = 'Paid') AND
        (s.route_no = route_rec.route_no) AND
        (EXTRACT(YEAR FROM s.departure_date_time) = In_SecondYear);

IF (v_secondYearAmount IS NULL) THEN
    v_secondYearAmount := 0.00;
END IF;

IF ((v_firstYearAmount IS NULL) AND (v_secondYearAmount = 0.00)) THEN
     v_firstYearAmount := 0.00;
     v_yearChange := 0;
ELSE
   IF ((v_firstYearAmount IS NULL) AND (v_secondYearAmount != 0.00)) THEN
        v_firstYearAmount := 0.00;
        v_yearChange := 100;
   ELSE
        v_yearChange := ((v_secondYearAmount - v_firstYearAmount)/v_firstYearAmount) * 100;
   END IF;
END IF;

DBMS_OUTPUT.PUT_Line(RPAD(route_rec.route_no,9,' ')||
                     RPAD(route_rec.departure_terminal,19,' ')||
                     RPAD(route_rec.destination_terminal,20,' ')||
                     LPAD('RM',3,' ')||
                     LPAD(TO_CHAR(v_firstYearAmount,'9,999,999.99'),13,' ')||
                     LPAD('RM',3,' ')||
                     LPAD(TO_CHAR(v_secondYearAmount,'9,999,999.99'),13,' ')||
                     LPAD(TO_CHAR(v_yearChange,'999.99'),8,' ')||'%')

v_firstYearTotalAmount := v_firstYearTotalAmount + v_firstYearAmount;
v_secondYearTotalAmount := v_secondYearTotalAmount + v_secondYearAmount;
v_loop := v_loop + 1;
```

```
  END LOOP;

  v_totalYearChange:=((v_secondYearTotalAmountv_firstYearTotalAmount)/v_firstYearTotalAmount)*100;

  IF (v_totalYearChange >=0) THEN
      v_status := 'Profit';
  ELSE
      v_status := 'Loss';
  END IF;

  DBMS_OUTPUT.PUT_LINE(''||LPAD(' ',41,' ')||'---------------'||' ---------------'||' --------');
  DBMS_OUTPUT.PUT_LINE('Total Yearly Sales Amount:'||
                       LPAD('RM',25,' ')||
                       LPAD(TO_CHAR(v_firstYearTotalAmount,'9,999,999.99'),13,' ')||
                       LPAD('RM',3,' ')||
                       LPAD(TO_CHAR(v_secondYearTotalAmount,'999,999.99'),13,' ')||
                       LPAD(TO_CHAR(v_totalYearChange,'999.99'),8,' ')||'%');
  DBMS_OUTPUT.PUT_LINE(''||LPAD(' ',41,' ')||'---------------'||' ---------------'||' --------');
  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('Profit/Loss compare year '||In_secondYear||' from year '||In_firstYear||
                       ':    '||RPAD(v_status,6,' '));
  DBMS_OUTPUT.PUT_LINE(chr(10));
  DBMS_OUTPUT.PUT_LINE('    '||LPAD(' ',29,' ')||'-End of report-');

EXCEPTION
   WHEN e_invalidInput THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Input: Input year must be a numeric.');
   WHEN e_invalidYear THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Input: Input year cannot be future year.');
   WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error Found. Please contact your Database Administrator');
END;
/
```

Sample Output:

```
SQL> exec proc_summaryReport('2013','2014')


                         Route Yearly Sales Summary Report
                         =================================


Printed Date: 27-11-2014                                                    Page:  1


Route No Departure Terminal Destination Terminal Year 2013 Sales Year 2014 Sales Changes
======== ================== ==================== =============== =============== ========
R10      Kuala Lumpur       Johor Bharu          RM   120,475.91 RM   109,428.52   -9.17%
R11      Kuala Lumpur       Georgetown           RM   119,130.37 RM   111,046.60   -6.79%
R12      Kuala Lumpur       Bandar Melaka        RM   119,463.56 RM   109,718.01   -8.16%
R13      Johor Bharu        Kuala Lumpur         RM   120,869.23 RM   110,417.19   -8.65%
R14      Georgetown         Kuala Lumpur         RM   119,056.93 RM   109,265.06   -8.22%
R15      Bandar Melaka      Kuala Lumpur         RM   121,736.36 RM   110,305.10   -9.39%
                                                 --------------- --------------- --------
Total Yearly Sales Amount:                       RM   720,732.36 RM   660,180.48   -8.40%
                                                 --------------- --------------- --------


Profit/Loss compare year 2014 from year 2013:    Loss


                                    -End of report-

PL/SQL procedure successfully completed.

SQL> spool off
```