

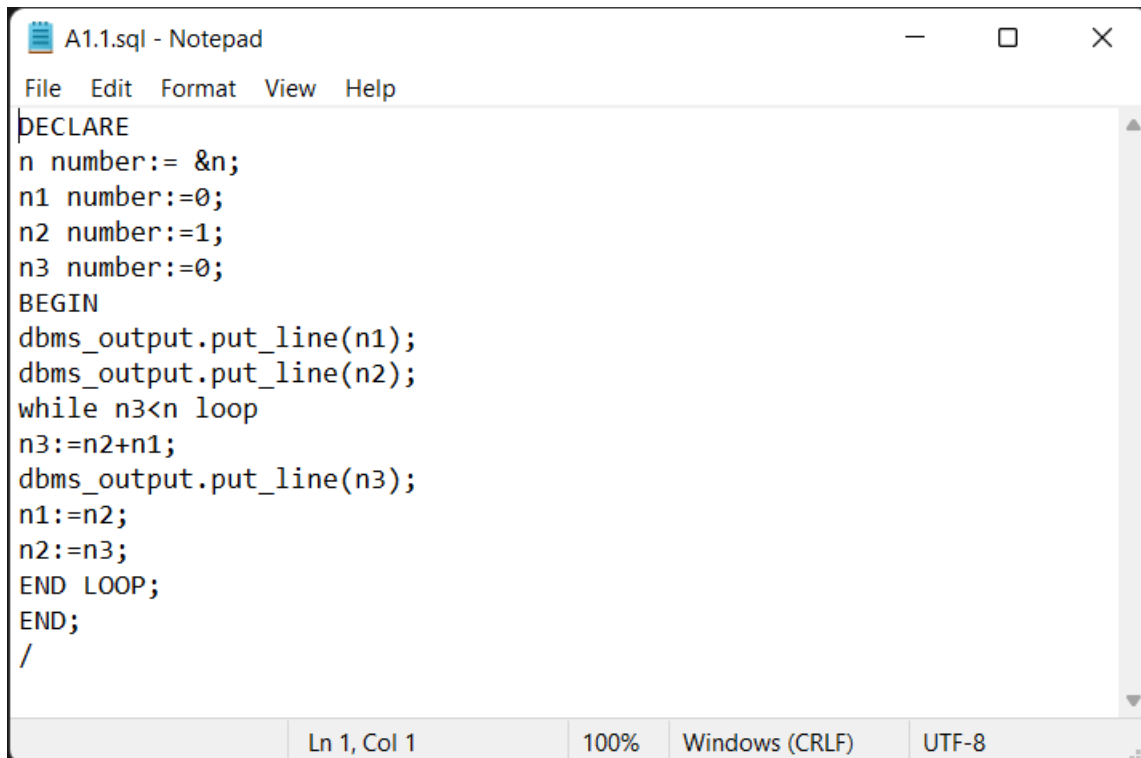
Exercise 4 – PL/SQL

21MCA0269

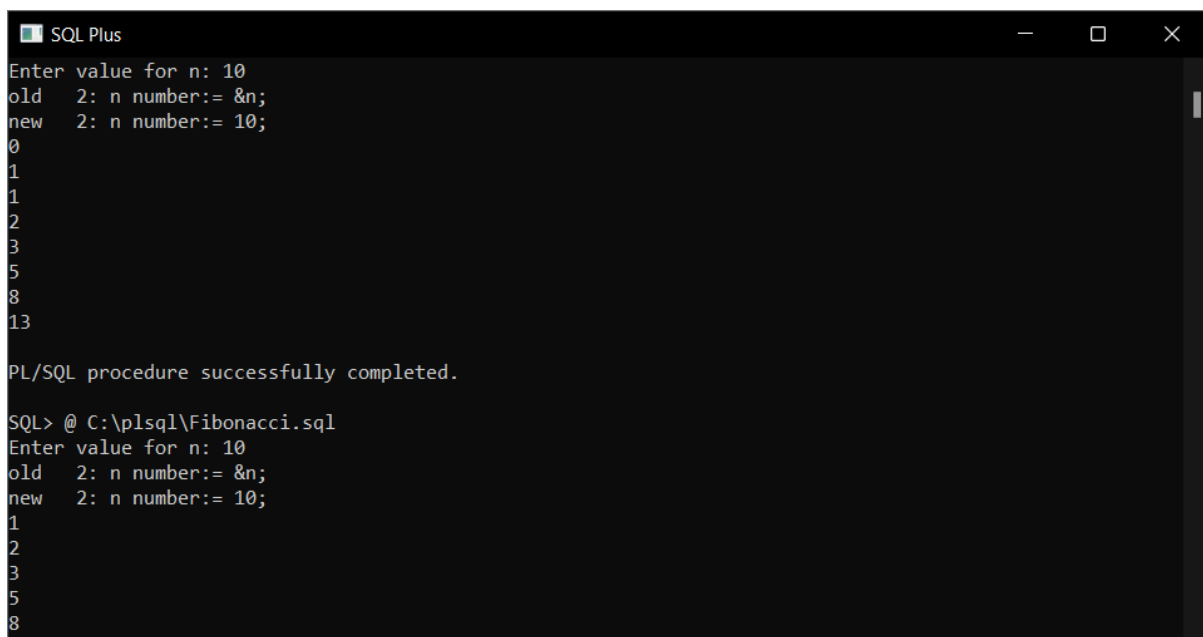
Slot: L5 + L6

PART A

1.1 – Fibonacci Series

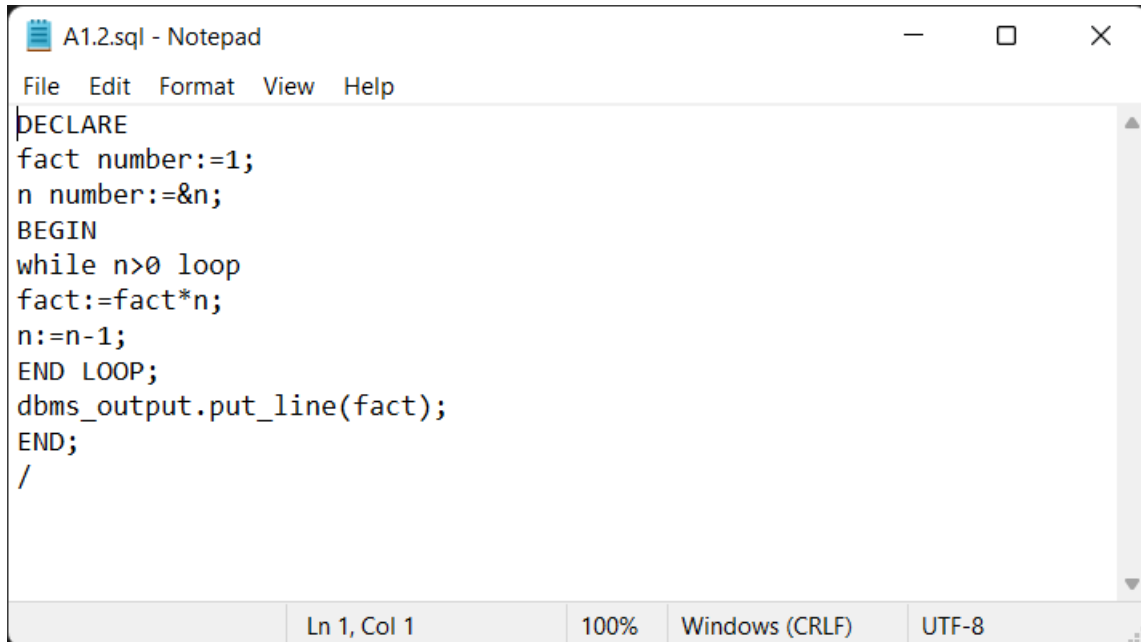


```
DECLARE
n number:= &n;
n1 number:=0;
n2 number:=1;
n3 number:=0;
BEGIN
dbms_output.put_line(n1);
dbms_output.put_line(n2);
while n3<n loop
n3:=n2+n1;
dbms_output.put_line(n3);
n1:=n2;
n2:=n3;
END LOOP;
END;
/
```



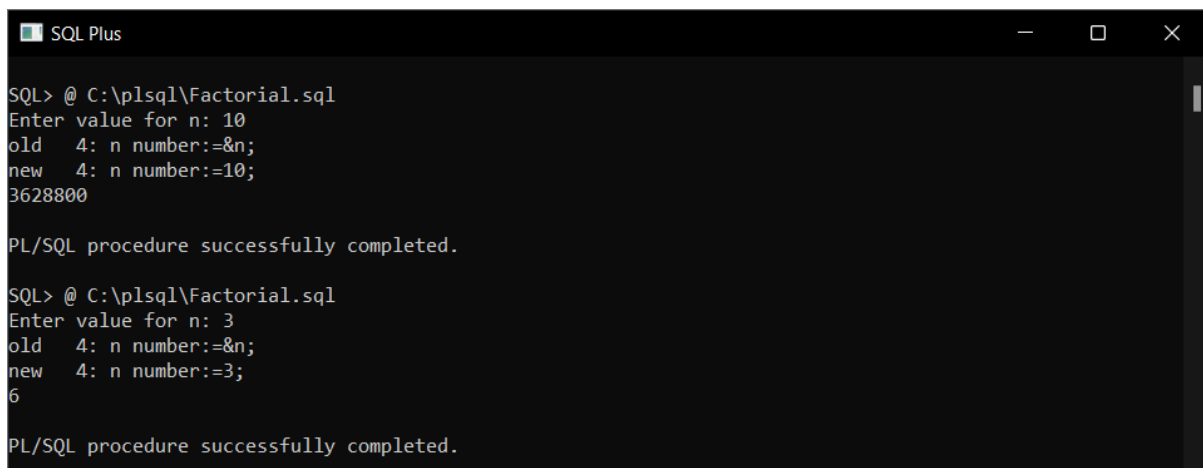
```
SQL Plus
Enter value for n: 10
old 2: n number:= &n;
new 2: n number:= 10;
0
1
1
2
3
5
8
13
PL/SQL procedure successfully completed.
SQL> @ C:\plsql\Fibonacci.sql
Enter value for n: 10
old 2: n number:= &n;
new 2: n number:= 10;
1
2
3
5
8
```

1.2 – Factorial of a number



```
DECLARE
fact number:=1;
n number:=&n;
BEGIN
while n>0 loop
fact:=fact*n;
n:=n-1;
END LOOP;
dbms_output.put_line(fact);
END;
/
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8



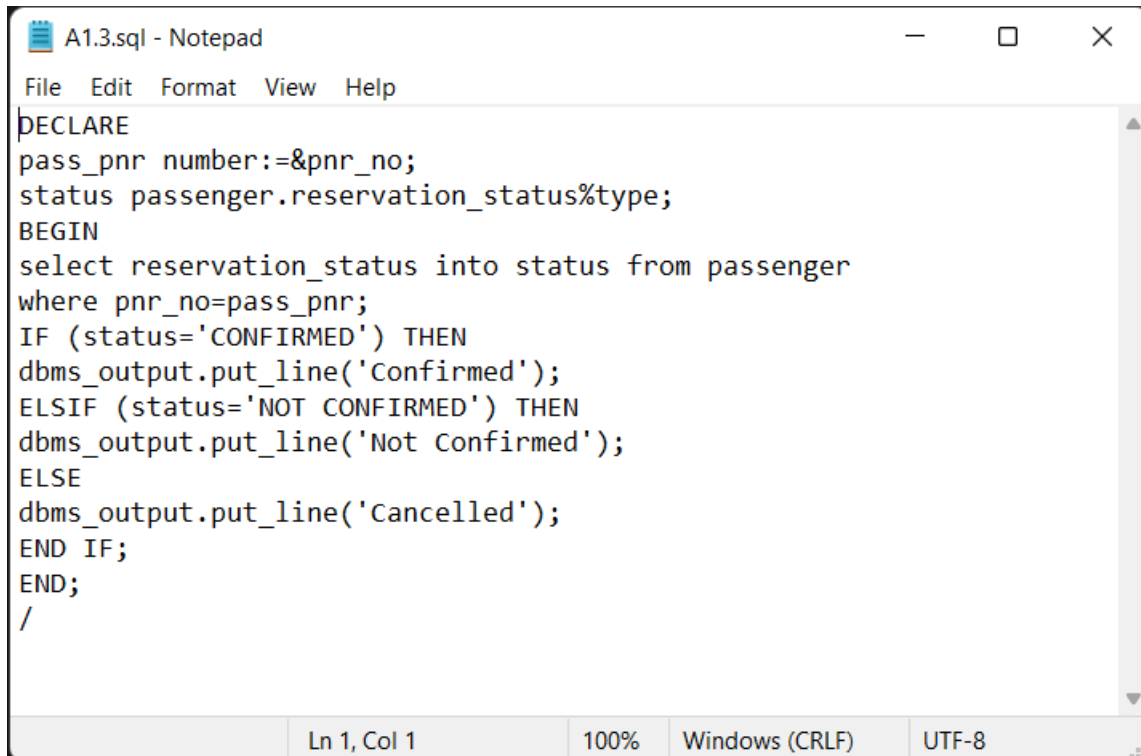
```
SQL> @ C:\plsql\Factorial.sql
Enter value for n: 10
old   4: n number:=&n;
new   4: n number:=10;
3628800

PL/SQL procedure successfully completed.

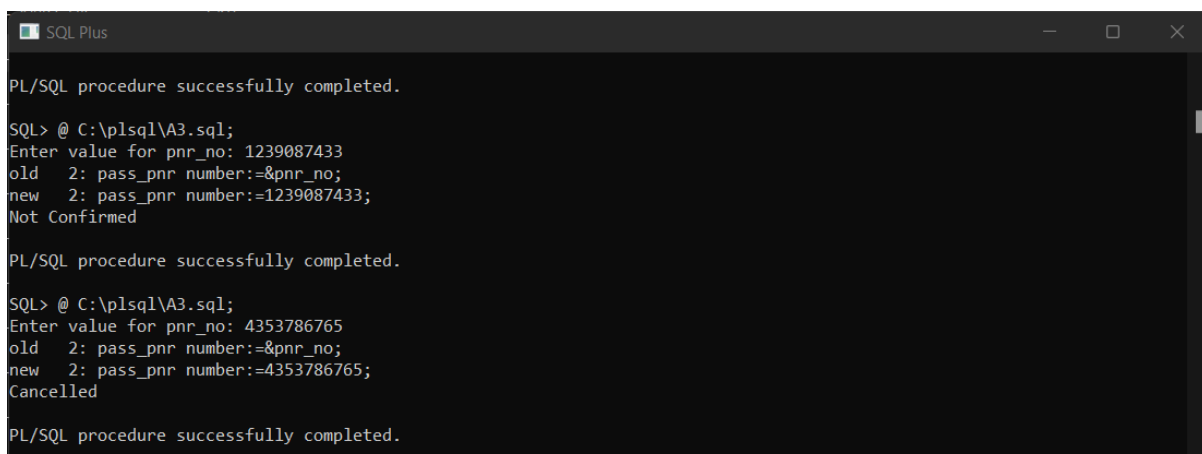
SQL> @ C:\plsql\Factorial.sql
Enter value for n: 3
old   4: n number:=&n;
new   4: n number:=3;
6

PL/SQL procedure successfully completed.
```

1.3) Print 'Not Confirmed' based on the reservation status, of a particular passenger.



```
A1.3.sql - Notepad
File Edit Format View Help
DECLARE
pass_pnr number:=&pnr_no;
status passenger.reservation_status%type;
BEGIN
select reservation_status into status from passenger
where pnr_no=pass_pnr;
IF (status='CONFIRMED') THEN
dbms_output.put_line('Confirmed');
ELSIF (status='NOT CONFIRMED') THEN
dbms_output.put_line('Not Confirmed');
ELSE
dbms_output.put_line('Cancelled');
END IF;
END;
/
Ln 1, Col 1    100%    Windows (CRLF)    UTF-8
```



```
SQL Plus
PL/SQL procedure successfully completed.

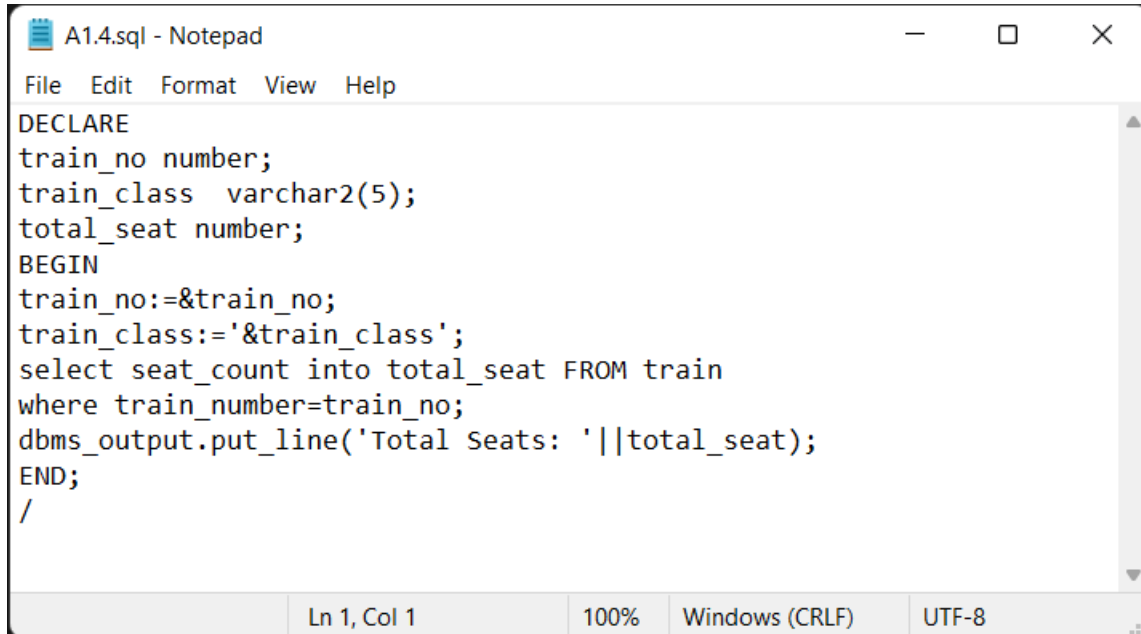
SQL> @ C:\pls1\A3.sql;
Enter value for pnr_no: 1239087433
old 2: pass_pnr number:=&pnr_no;
new 2: pass_pnr number:=1239087433;
Not Confirmed

PL/SQL procedure successfully completed.

SQL> @ C:\pls1\A3.sql;
Enter value for pnr_no: 4353786765
old 2: pass_pnr number:=&pnr_no;
new 2: pass_pnr number:=4353786765;
Cancelled

PL/SQL procedure successfully completed.
```

1.4) Print the total no of available seats for a particular train and for a particular class.



```
DECLARE
train_no number;
train_class varchar2(5);
total_seat number;
BEGIN
train_no:=&train_no;
train_class:='&train_class';
select seat_count into total_seat FROM train
where train_number=train_no;
dbms_output.put_line('Total Seats: '||total_seat);
END;
/
```

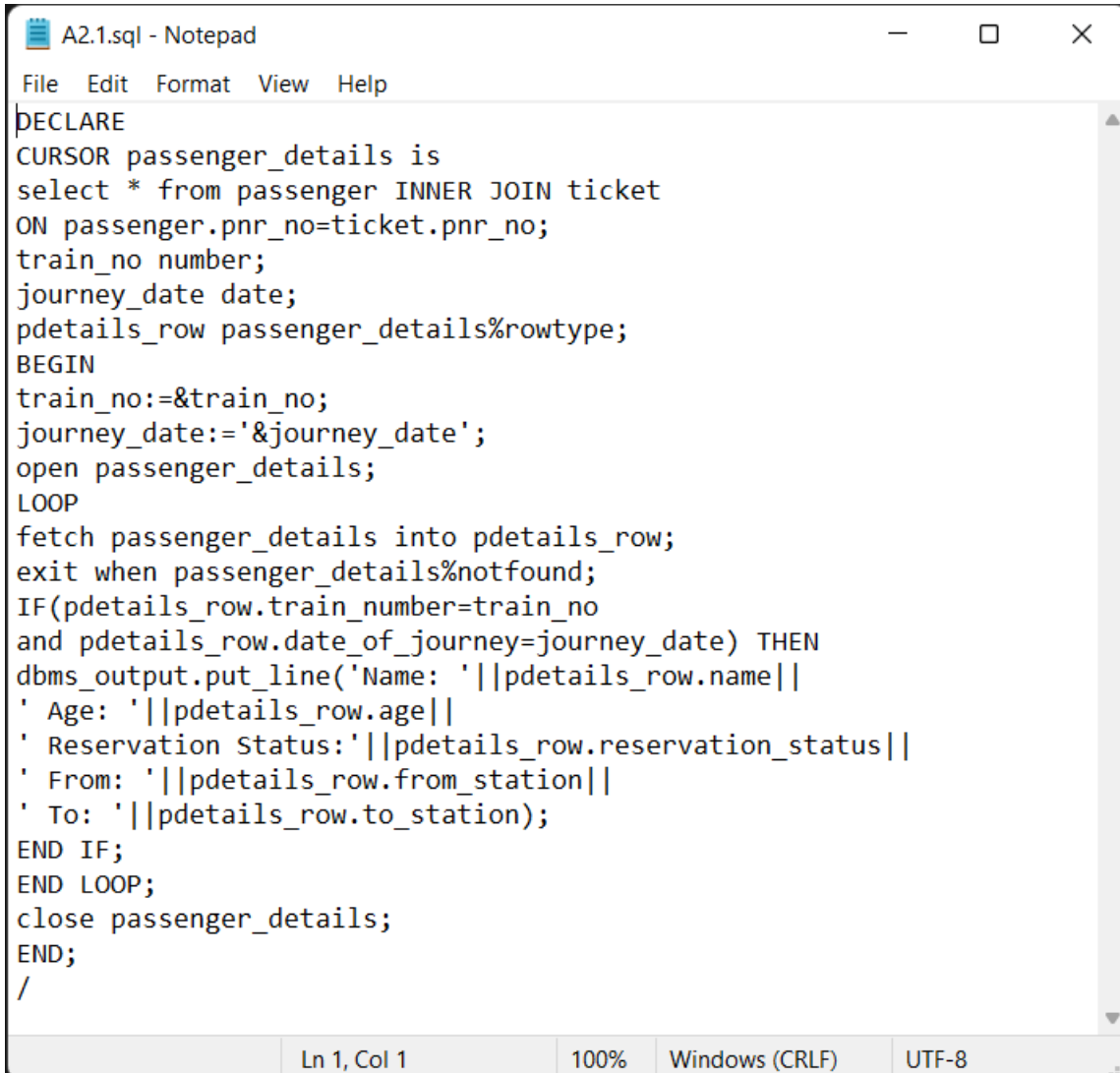


```
SQL> @ C:\plsql\A1.4.sql
Enter value for train_no: 11028
old 6: train_no:=&train_no;
new 6: train_no:=11028;
Enter value for train_class: SL
old 7: train_class:=&train_class';
new 7: train_class:='SL';
Total Seats: 95

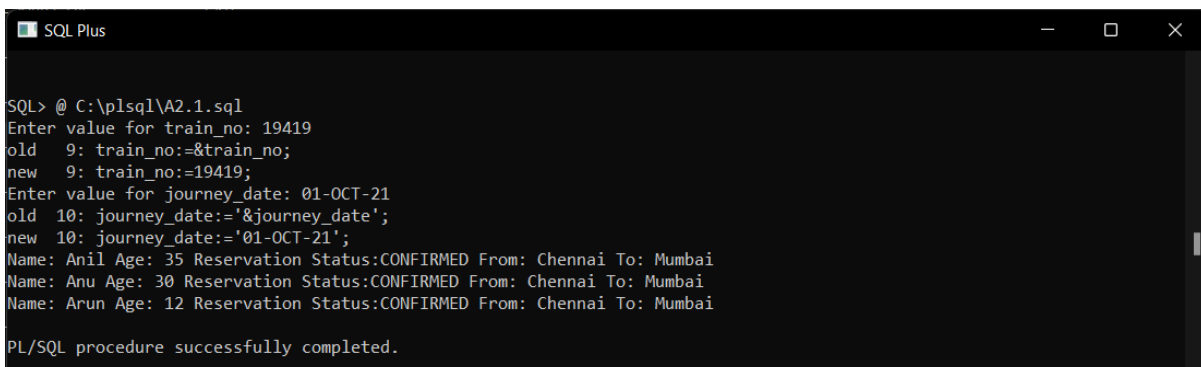
PL/SQL procedure successfully completed.
```

Cursors

2.1) Retrieve the passenger details for 'X' train no and given journey date.

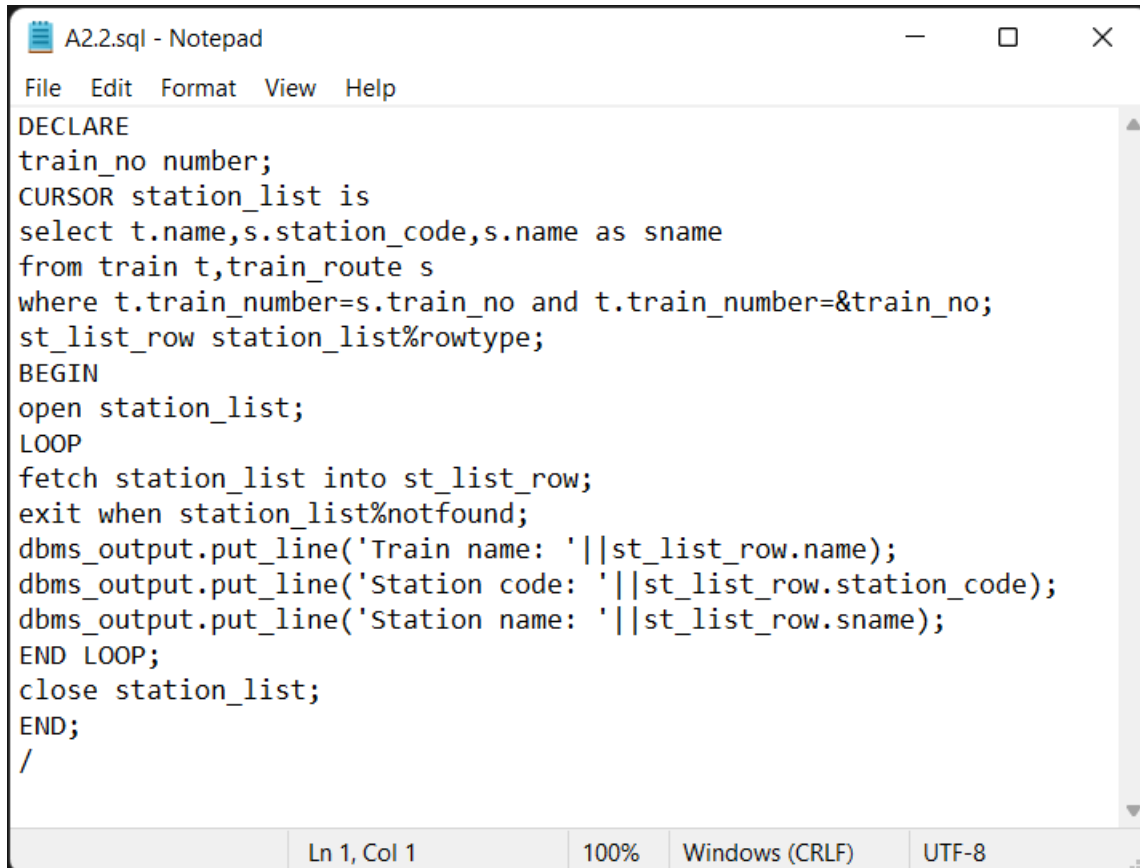


```
DECLARE
CURSOR passenger_details is
select * from passenger INNER JOIN ticket
ON passenger.pnr_no=ticket.pnr_no;
train_no number;
journey_date date;
pdetails_row passenger_details%rowtype;
BEGIN
train_no:=&train_no;
journey_date:='&journey_date';
open passenger_details;
LOOP
fetch passenger_details into pdetails_row;
exit when passenger_details%notfound;
IF(pdetails_row.train_number=train_no
and pdetails_row.date_of_journey=journey_date) THEN
dbms_output.put_line('Name: '||pdetails_row.name||
' Age: '||pdetails_row.age||
' Reservation Status:'||pdetails_row.reservation_status||
' From: '||pdetails_row.from_station||
' To: '||pdetails_row.to_station);
END IF;
END LOOP;
close passenger_details;
END;
/
```



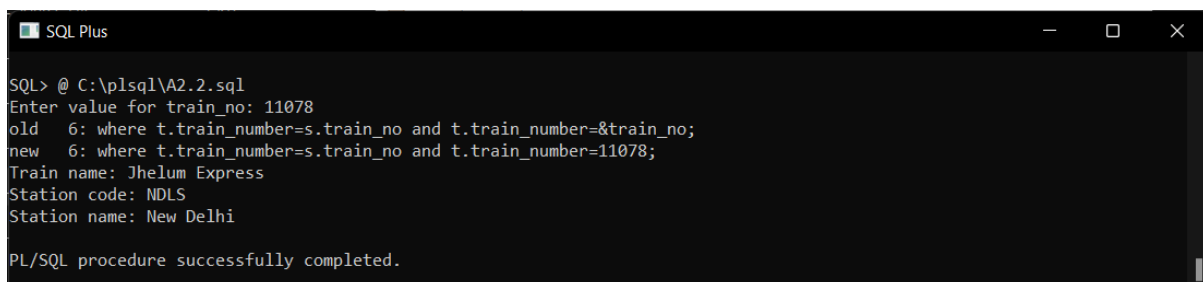
```
SQL> @ C:\p1sql\A2.1.sql
Enter value for train_no: 19419
old 9: train_no:=&train_no;
new 9: train_no:=19419;
Enter value for journey_date: 01-OCT-21
old 10: journey_date:=&journey_date';
new 10: journey_date:='01-OCT-21';
Name: Anil Age: 35 Reservation Status:CONFIRMED From: Chennai To: Mumbai
Name: Anu Age: 30 Reservation Status:CONFIRMED From: Chennai To: Mumbai
Name: Arun Age: 12 Reservation Status:CONFIRMED From: Chennai To: Mumbai
PL/SQL procedure successfully completed.
```

2.2) Display the train name (once) and the substation names.



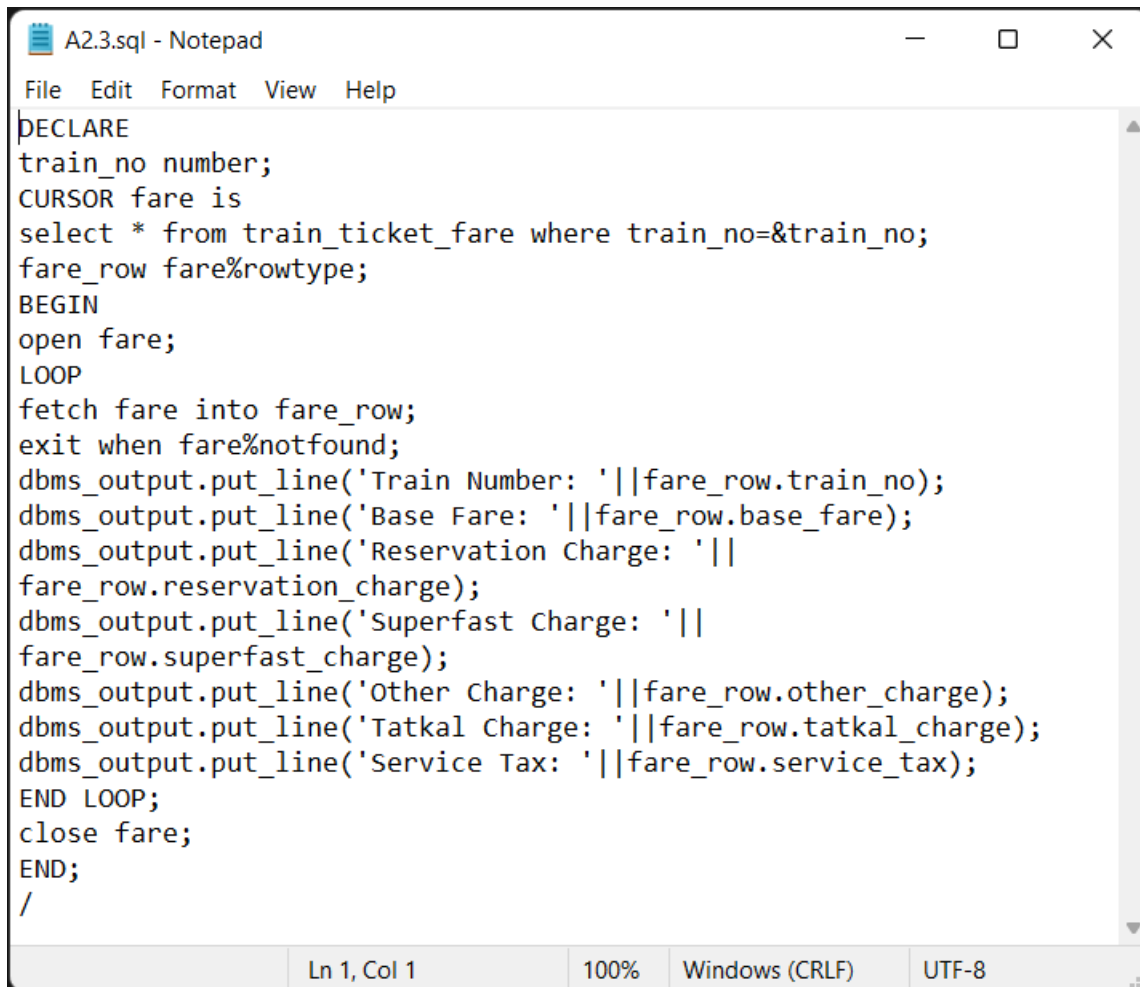
```
A2.2.sql - Notepad
File Edit Format View Help
DECLARE
train_no number;
CURSOR station_list is
select t.name,s.station_code,s.name as sname
from train t,train_route s
where t.train_number=s.train_no and t.train_number=&train_no;
st_list_row station_list%rowtype;
BEGIN
open station_list;
LOOP
fetch station_list into st_list_row;
exit when station_list%notfound;
dbms_output.put_line('Train name: '||st_list_row.name);
dbms_output.put_line('Station code: '||st_list_row.station_code);
dbms_output.put_line('Station name: '||st_list_row.sname);
END LOOP;
close station_list;
END;
/

Ln 1, Col 1    100%    Windows (CRLF)    UTF-8
```



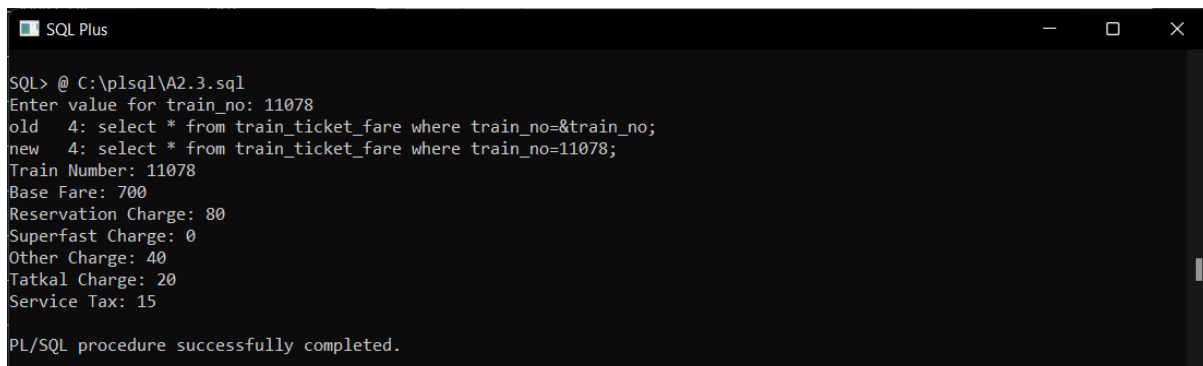
```
SQL Plus
SQL> @ C:\plsql\A2.2.sql
Enter value for train_no: 11078
old   6: where t.train_number=s.train_no and t.train_number=&train_no;
new   6: where t.train_number=s.train_no and t.train_number=11078;
Train name: Jhelum Express
Station code: NDLS
Station name: New Delhi
PL/SQL procedure successfully completed.
```

2.3) Display the fare details of a particular train (use basic exception).



```
A2.3.sql - Notepad
File Edit Format View Help
DECLARE
train_no number;
CURSOR fare is
select * from train_ticket_fare where train_no=&train_no;
fare_row fare%rowtype;
BEGIN
open fare;
LOOP
fetch fare into fare_row;
exit when fare%notfound;
dbms_output.put_line('Train Number: '||fare_row.train_no);
dbms_output.put_line('Base Fare: '||fare_row.base_fare);
dbms_output.put_line('Reservation Charge: '||
fare_row.reservation_charge);
dbms_output.put_line('Superfast Charge: '||
fare_row.superfast_charge);
dbms_output.put_line('Other Charge: '||fare_row.other_charge);
dbms_output.put_line('Tatkal Charge: '||fare_row.tatkal_charge);
dbms_output.put_line('Service Tax: '||fare_row.service_tax);
END LOOP;
close fare;
END;
/

Ln 1, Col 1      100%   Windows (CRLF)   UTF-8
```



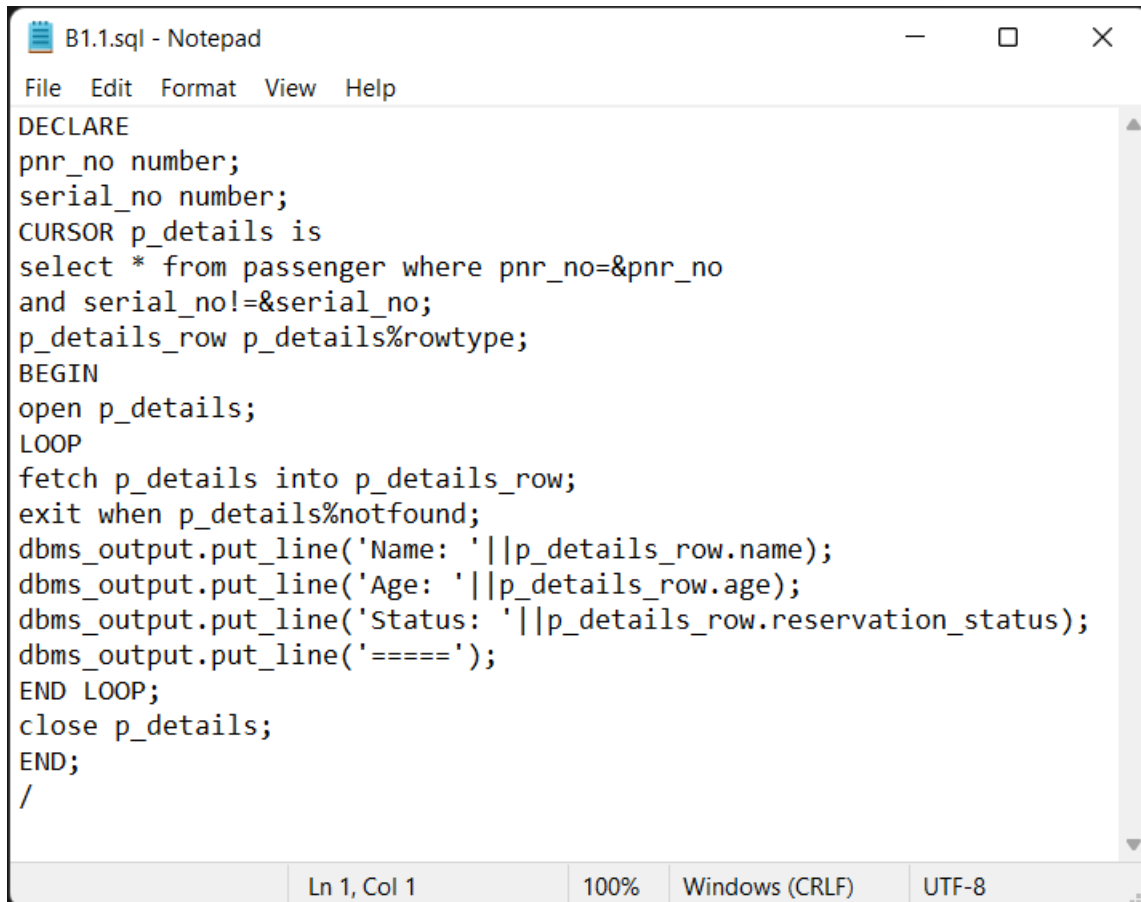
```
SQL Plus
SQL> @ C:\plsql\A2.3.sql
Enter value for train_no: 11078
old 4: select * from train_ticket_fare where train_no=&train_no;
new 4: select * from train_ticket_fare where train_no=11078;
Train Number: 11078
Base Fare: 700
Reservation Charge: 80
Superfast Charge: 0
Other Charge: 40
Tatkal Charge: 20
Service Tax: 15

PL/SQL procedure successfully completed.
```

End of PART A

PART B

1.1) List the details of the passengers who has reserved next to “Mr. X”.



A screenshot of a Notepad window titled "B1.1.sql - Notepad". The window contains a PL/SQL script designed to find passengers who have reserved next to a specific passenger (Mr. X). The script declares two variables, pnr_no and serial_no, and a cursor p_details. It then opens the cursor, loops through the results, and prints the name, age, and reservation status of each passenger. The script ends with a slash (/).

```
DECLARE
pnr_no number;
serial_no number;
CURSOR p_details is
select * from passenger where pnr_no=&pnr_no
and serial_no!=&serial_no;
p_details_row p_details%rowtype;
BEGIN
open p_details;
LOOP
fetch p_details into p_details_row;
exit when p_details%notfound;
dbms_output.put_line('Name: '||p_details_row.name);
dbms_output.put_line('Age: '||p_details_row.age);
dbms_output.put_line('Status: '||p_details_row.reservation_status);
dbms_output.put_line('====');
END LOOP;
close p_details;
END;
/
```

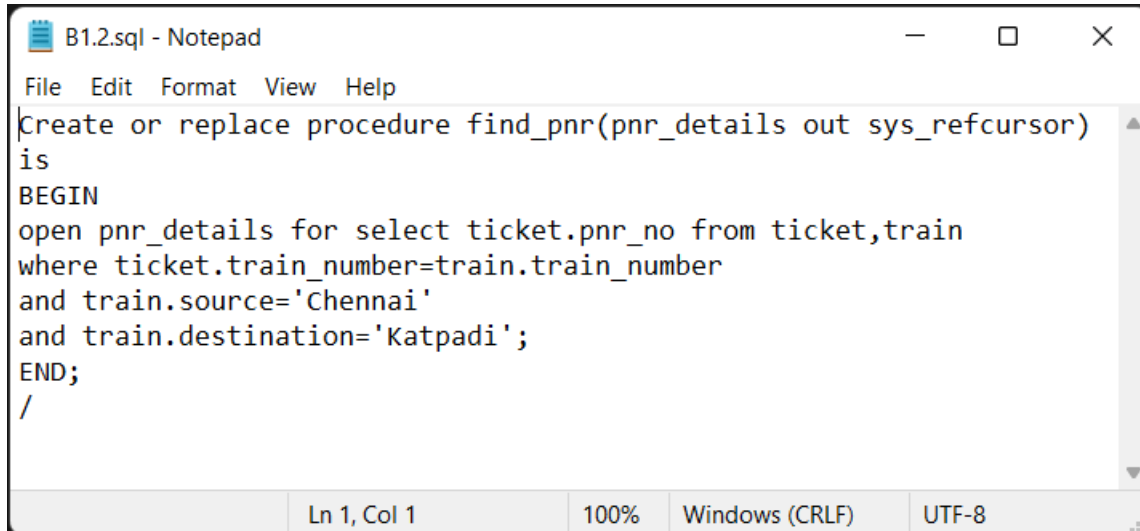
The status bar at the bottom of the Notepad window shows "Ln 1, Col 1", "100%", "Windows (CRLF)", and "UTF-8".



A screenshot of a SQL Plus window titled "SQL Plus". The window shows the execution of the PL/SQL script from the previous image. It prompts for the pnr_no and serial_no values, and then displays the output of the script, which lists the details of the passengers who have reserved next to Mr. X. The output shows two passengers: Anil (Age: 35, Status: CONFIRMED) and Arun (Age: 12, Status: CONFIRMED). The script is successfully completed.

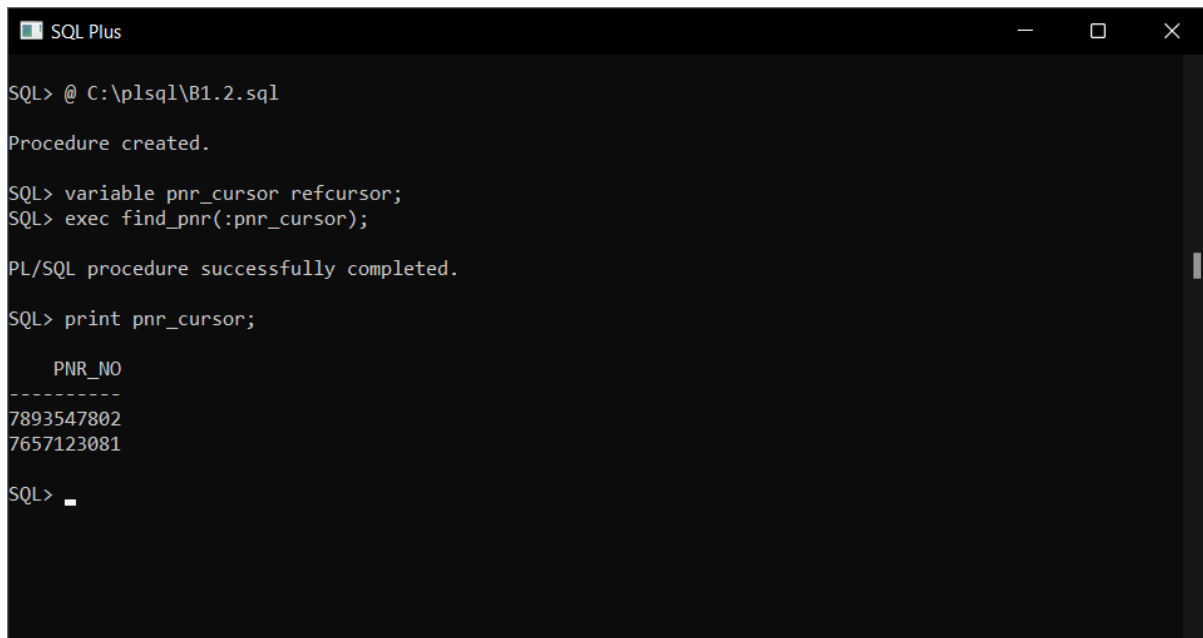
```
SQL> @ C:\plsql\B1.1.sql
Enter value for pnr_no: 3456345767
old 5: select * from passenger where pnr_no=&pnr_no
new 5: select * from passenger where pnr_no=3456345767
Enter value for serial_no: 2
old 6: and serial_no!=&serial_no;
new 6: and serial_no!=2;
Name: Anil
Age: 35
Status: CONFIRMED
=====
Name: Arun
Age: 12
Status: CONFIRMED
=====
PL/SQL procedure successfully completed.
```


1.2) PNR number of passengers for a given source and a destination.



A screenshot of a Notepad window titled "B1.2.sql - Notepad". The window contains a PL/SQL procedure definition. The menu bar includes "File", "Edit", "Format", "View", and "Help". The status bar at the bottom shows "Ln 1, Col 1", "100%", "Windows (CRLF)", and "UTF-8".

```
Create or replace procedure find_pnr(pnr_details out sys_refcursor)
is
BEGIN
open pnr_details for select ticket.pnr_no from ticket,train
where ticket.train_number=train.train_number
and train.source='Chennai'
and train.destination='Katpadi';
END;
/
```



A screenshot of an SQL Plus window titled "SQL Plus". The window shows the execution of the PL/SQL procedure defined in the previous screenshot. The prompt "SQL>" is used to enter commands. The output shows the procedure was created successfully, and the results of the query are displayed in a table format.

```
SQL> @ C:\plsql\B1.2.sql

Procedure created.

SQL> variable pnr_cursor refcursor;
SQL> exec find_pnr(:pnr_cursor);

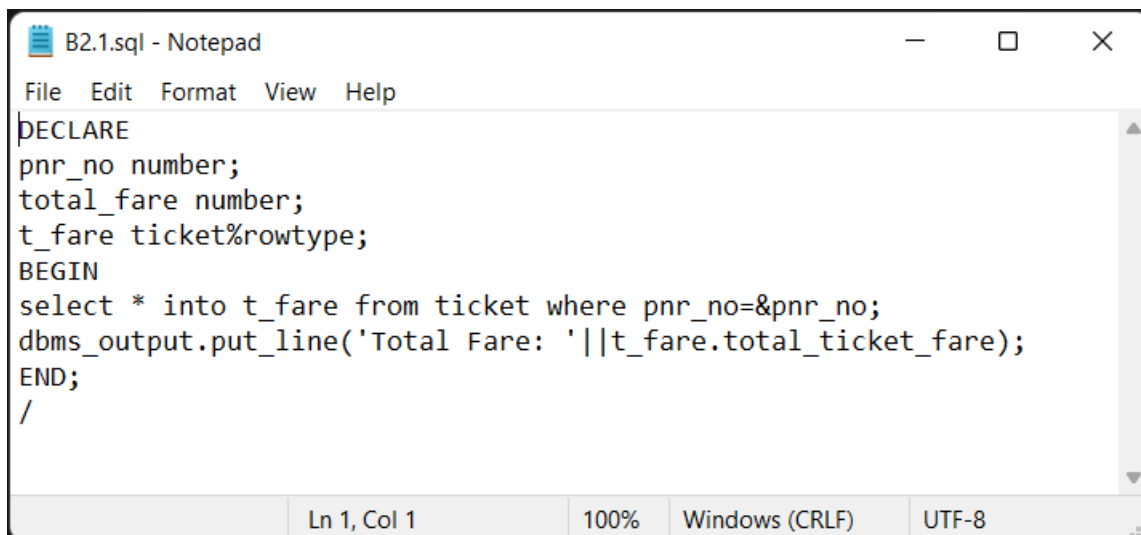
PL/SQL procedure successfully completed.

SQL> print pnr_cursor;

      PNR_NO
-----
7893547802
7657123081

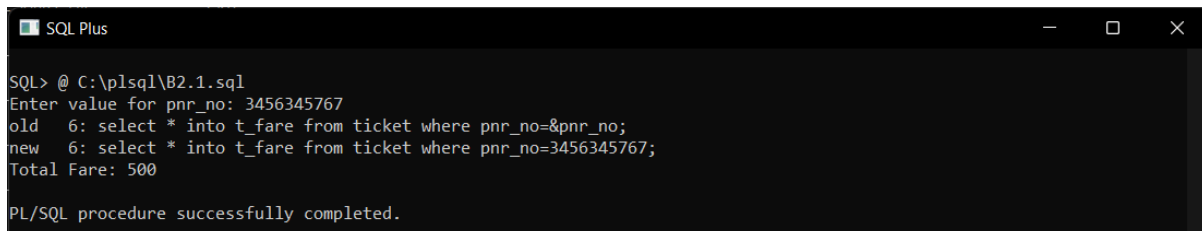
SQL> _
```

2.1) Get the PNR number and return the total ticket fare.



```
B2.1.sql - Notepad
File Edit Format View Help
DECLARE
pnr_no number;
total_fare number;
t_fare ticket%rowtype;
BEGIN
select * into t_fare from ticket where pnr_no=&pnr_no;
dbms_output.put_line('Total Fare: '||t_fare.total_ticket_fare);
END;
/

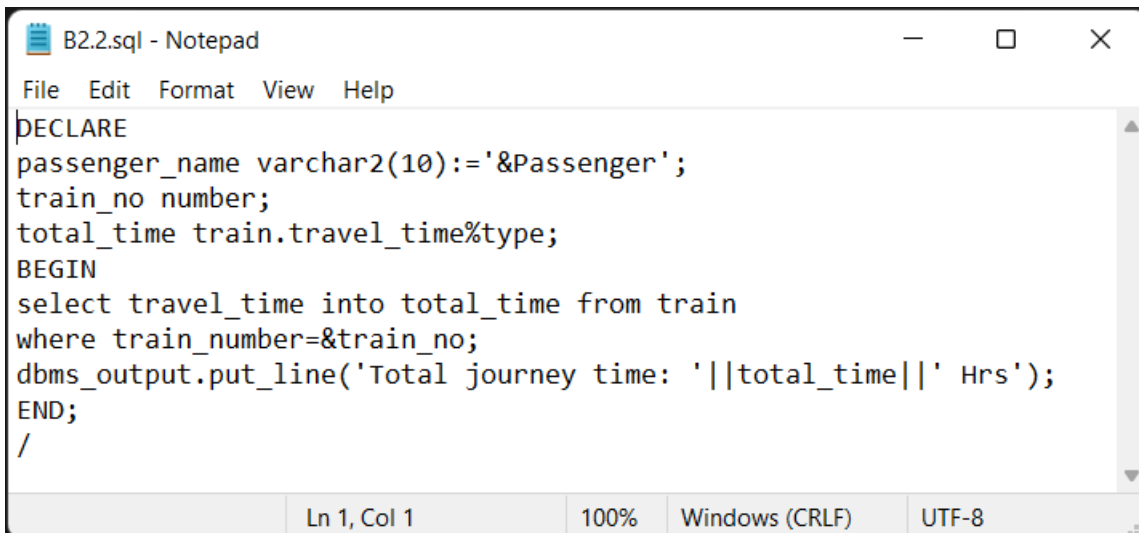
Ln 1, Col 1    100%    Windows (CRLF)    UTF-8
```



```
SQL Plus
SQL> @ C:\plsql\B2.1.sql
Enter value for pnr_no: 3456345767
old 6: select * into t_fare from ticket where pnr_no=&pnr_no;
new 6: select * into t_fare from ticket where pnr_no=3456345767;
Total Fare: 500

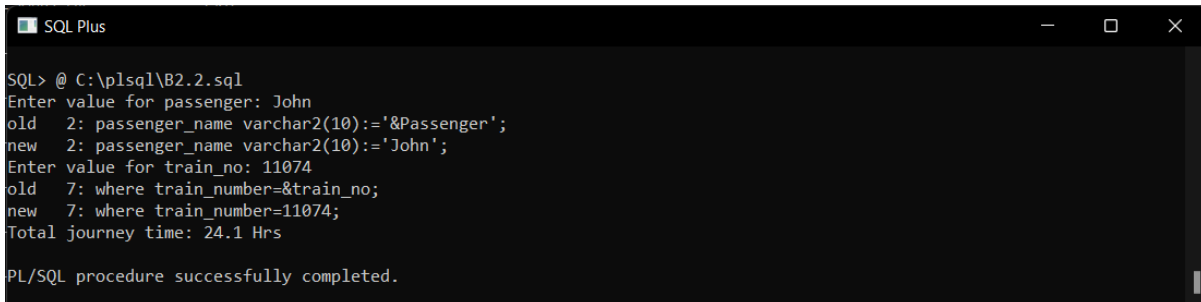
PL/SQL procedure successfully completed.
```

2.2) Get the Passenger name, train no and return the total journey time in hours and minutes.



```
File Edit Format View Help
DECLARE
passenger_name varchar2(10):='&Passenger';
train_no number;
total_time train.travel_time%type;
BEGIN
select travel_time into total_time from train
where train_number=&train_no;
dbms_output.put_line('Total journey time: '||total_time||' Hrs');
END;
/

Ln 1, Col 1      100%   Windows (CRLF)   UTF-8
```



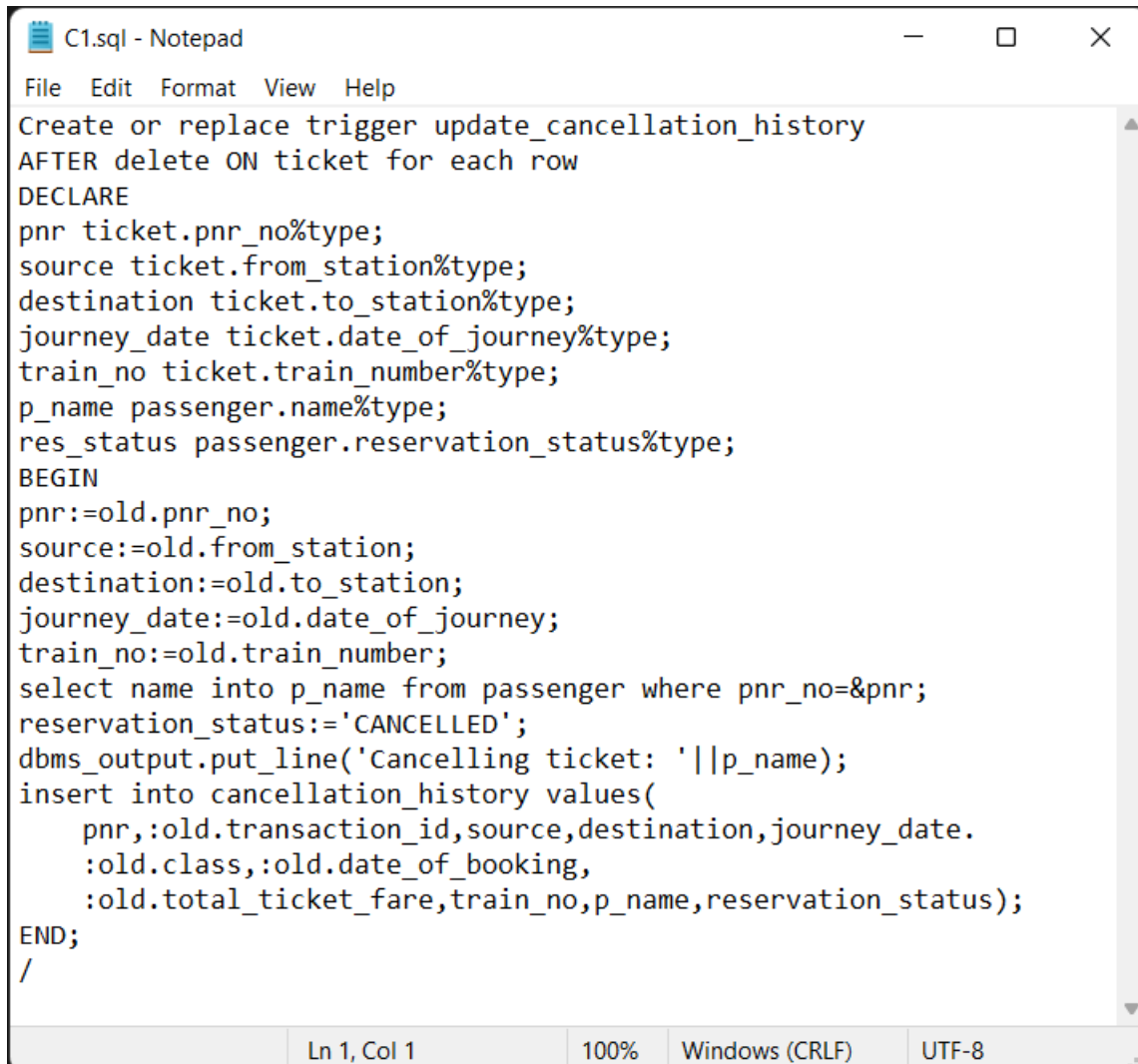
```
SQL Plus
SQL> @ C:\plsql\B2.2.sql
Enter value for passenger: John
old 2: passenger_name varchar2(10):='&Passenger';
new 2: passenger_name varchar2(10):='John';
Enter value for train_no: 11074
old 7: where train_number=&train_no;
new 7: where train_number=11074;
Total journey time: 24.1 Hrs

PL/SQL procedure successfully completed.
```

End of PART B

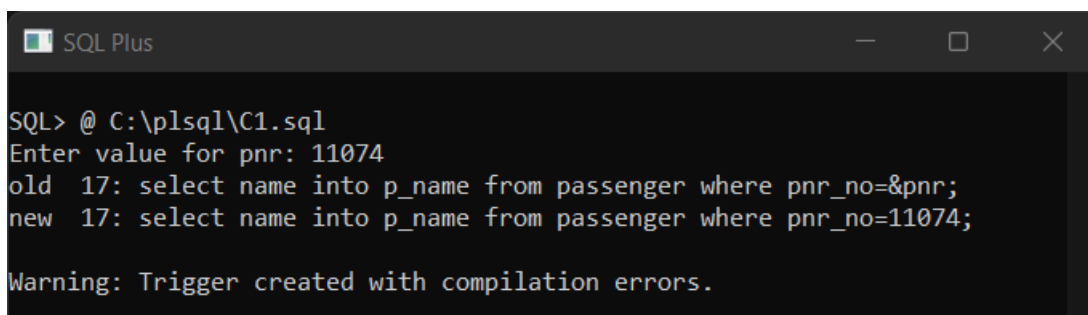
PART C - Triggers

1) When a passenger cancels a ticket, do the necessary process and update the cancellation history table.



A screenshot of a Notepad window titled "C1.sql - Notepad". The window contains SQL code for creating or replacing a trigger named "update_cancellation_history". The trigger is defined as "AFTER delete ON ticket for each row". The code declares several variables: "pnr", "source", "destination", "journey_date", "train_no", "p_name", and "res_status". It then begins a block of code that assigns values to these variables from the "old" record, selects the passenger name, updates the reservation status to "CANCELLED", outputs a message, and inserts a record into the "cancellation_history" table. The code ends with "END;" and a forward slash. The status bar at the bottom shows "Ln 1, Col 1", "100%", "Windows (CRLF)", and "UTF-8".

```
File Edit Format View Help
Create or replace trigger update_cancellation_history
AFTER delete ON ticket for each row
DECLARE
pnr ticket.pnr_no%type;
source ticket.from_station%type;
destination ticket.to_station%type;
journey_date ticket.date_of_journey%type;
train_no ticket.train_number%type;
p_name passenger.name%type;
res_status passenger.reservation_status%type;
BEGIN
pnr:=old.pnr_no;
source:=old.from_station;
destination:=old.to_station;
journey_date:=old.date_of_journey;
train_no:=old.train_number;
select name into p_name from passenger where pnr_no=&pnr;
reservation_status:='CANCELLED';
dbms_output.put_line('Cancelling ticket: '||p_name);
insert into cancellation_history values(
    pnr,:old.transaction_id,source,destination,journey_date.
    :old.class,:old.date_of_booking,
    :old.total_ticket_fare,train_no,p_name,reservation_status);
END;
/
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

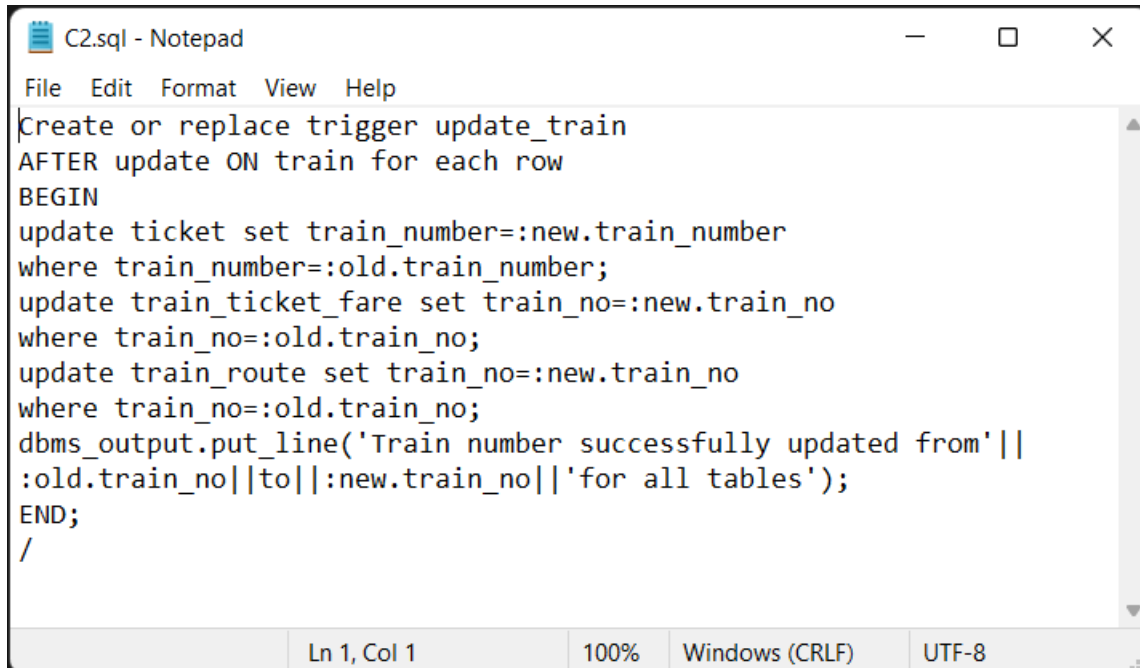


A screenshot of a SQL Plus window titled "SQL Plus". The window shows the execution of the SQL code from the previous screenshot. It prompts for the value of "pnr" and shows the execution of the trigger. The output shows the passenger name being selected and the reservation status being updated. A warning message at the bottom states "Warning: Trigger created with compilation errors.".

```
SQL> @ C:\plsql\C1.sql
Enter value for pnr: 11074
old 17: select name into p_name from passenger where pnr_no=&pnr;
new 17: select name into p_name from passenger where pnr_no=11074;

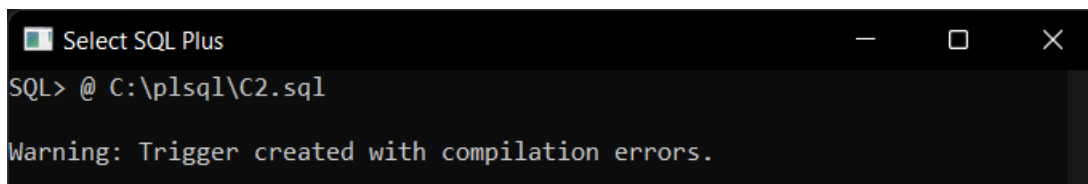
Warning: Trigger created with compilation errors.
```

2) When train number is changed, update it in referencing tables.



```
File Edit Format View Help
Create or replace trigger update_train
AFTER update ON train for each row
BEGIN
update ticket set train_number=:new.train_number
where train_number=:old.train_number;
update train_ticket_fare set train_no=:new.train_no
where train_no=:old.train_no;
update train_route set train_no=:new.train_no
where train_no=:old.train_no;
dbms_output.put_line('Train number successfully updated from'||
:old.train_no||to||:new.train_no||'for all tables');
END;
/
```

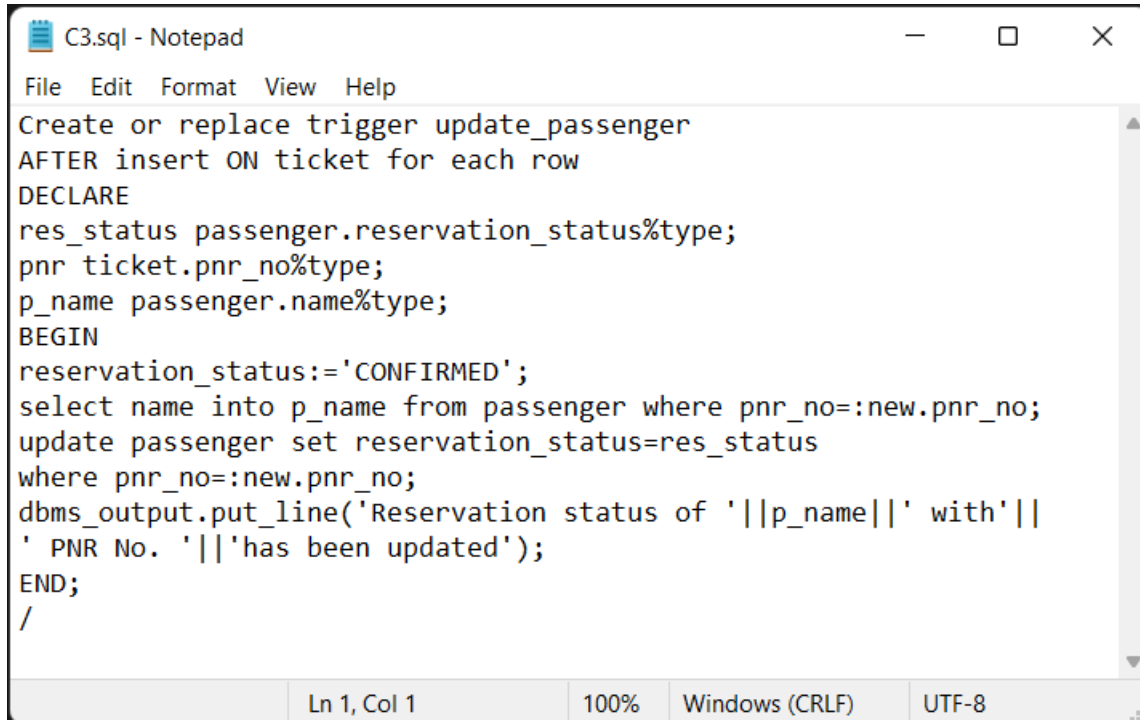
Ln 1, Col 1 100% Windows (CRLF) UTF-8



```
Select SQL Plus
SQL> @ C:\plsql\C2.sql

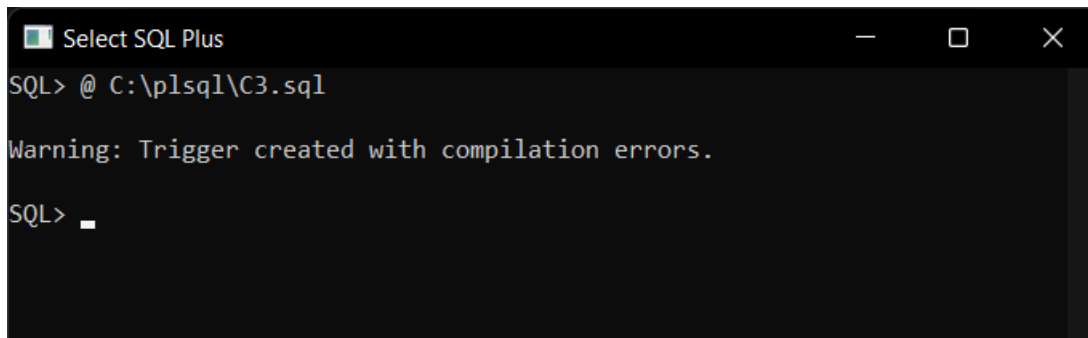
Warning: Trigger created with compilation errors.
```

3) When a passenger record is inserted reservation status should be automatically updated.



```
C3.sql - Notepad
File Edit Format View Help
Create or replace trigger update_passenger
AFTER insert ON ticket for each row
DECLARE
res_status passenger.reservation_status%type;
pnr ticket.pnr_no%type;
p_name passenger.name%type;
BEGIN
reservation_status:='CONFIRMED';
select name into p_name from passenger where pnr_no=:new.pnr_no;
update passenger set reservation_status=res_status
where pnr_no=:new.pnr_no;
dbms_output.put_line('Reservation status of '||p_name||' with'||
' PNR No. '||'has been updated');
END;
/
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8



```
Select SQL Plus
SQL> @ C:\plsql\C3.sql
Warning: Trigger created with compilation errors.
SQL> _
```

- END OF ASSIGNMENT -