ITMD 523 *Advanced Topics in Data Management* HW 7

Student Name	**Sivaranjani Prabasankar**	Section	
Instructor	***Luke Papademas***	Due Date	**04/13/ 2019**

Part	1	2	3	4	**TOTAL**	**Score**
Maximum Points	25 points	25 points	25 points	25 points	**100** points	

Textbook Reading Assignment

Thoroughly read Week 1 - 14 course lecture notes.

For the exercises in this assignment, please examine the data sheets and SQL code which follows:

Table: Books

BOOK_ID	TITLE	Author_Last_Name	Author_First_Name	CLASSIFY
1000	The Picture of Dorian Gray	Wilde	Oscar	fiction
1001	Think Big	Carson	Dr. Ben	non - fiction
1003	Mathematical Scandals	Pappas		non - fiction
1004	SQL	Harris	Andy	non - fiction
1010	Excel 2016	Chan		non - fiction

Table: Patrons

PATRON_ID	LAST_NAME	ZIP	PURCHASES
101	Wang	60616	$0.00
102	Peters	60605	$9.95
103	Wang	48204	$58.98
104	Ahmed	60631	$0.00
105	Nicholas	48204	$48.95

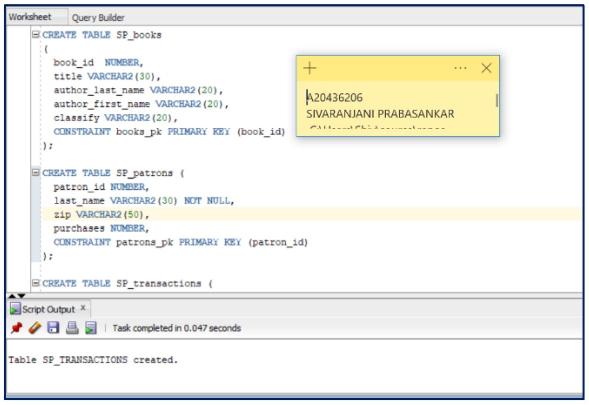
Table: Transactions

TRANSACTION_ID	PATRON_ID	BOOK_ID	TRANSACTION_DATE	PRICE
10	105	1000	11-MAR-18	$19.95
11	102	1003	12-MAR-18	$9.95
12	103	1004	12-MAR-18	$39.49
13	103	1010	16-MAR-18	$19.49
14	105	1004	19-MAR-18	$29.00

```
--drop table books;
CREATE TABLE books
(
  book_id  NUMBER,
  title VARCHAR2(30),
  author_last_name VARCHAR2(20),
  author_first_name VARCHAR2(20),
  classify VARCHAR2(20),
  CONSTRAINT books_pk PRIMARY KEY (book_id)
);
```

```
INSERT INTO books(book_id, title, author_last_name, author_first_name,
classify)
VALUES (1000, 'The Picture of Dorian Gray', 'Wilde', 'Oscar',
'fiction');
INSERT INTO books(book_id, title, author_last_name, author_first_name,
classify)
VALUES (1001, 'Think Big', 'Carson', 'Dr. Ben', 'non - fiction');
INSERT INTO books(book_id, title, author_last_name, author_first_name,
classify)
VALUES (1003, 'Mathematical Scandals', 'Pappas', ' ', 'non - fiction');
INSERT INTO books(book_id, title, author_last_name, author_first_name,
classify)
VALUES (1004, 'SQL', 'Harris', 'Andy', 'non - fiction');
INSERT INTO books(book_id, title, author_last_name, author_first_name,
classify)
VALUES (1010, 'Excel 2016', 'Chan', ' ', 'non - fiction');
commit;
--drop table patrons;
CREATE TABLE patrons (
  patron_id NUMBER,
  last_name VARCHAR2(30) NOT NULL,
  zip VARCHAR2(50),
  purchases NUMBER,
  CONSTRAINT patrons_pk PRIMARY KEY (patron_id)
);
INSERT INTO patrons(patron_id, last_name, zip, purchases)
VALUES(101, 'Wang', 60616, 0.00);
INSERT INTO patrons(patron_id, last_name, zip, purchases)
VALUES(102, 'Peters', 60605, 9.95);
INSERT INTO patrons(patron_id, last_name, zip, purchases)
VALUES(103, 'Wang', 48204, 58.98);
INSERT INTO patrons(patron_id, last_name, zip, purchases)
VALUES(104, 'Ahmed', 60631, 0.00);
INSERT INTO patrons(patron_id, last_name, zip, purchases)
VALUES(105, 'Nicholas', 48204, 48.95);
commit;
--drop table transactions;
CREATE TABLE transactions (
  transaction_id NUMBER,
  patron_id CONSTRAINT for_key_patron_id
  REFERENCES patrons(patron_id),
  book_id CONSTRAINT for_key_book_id
  REFERENCES books(book_id),
  transaction_date DATE NOT NULL,
  price NUMBER,
  CONSTRAINT transactions_pk PRIMARY KEY (transaction_id)
);
```

```
INSERT INTO transactions(transaction_id, patron_id, book_id,
transaction_date, price)
VALUES(10, 105, 1000, '11-Mar-2018', 19.95);
INSERT INTO transactions(transaction_id, patron_id, book_id,
transaction_date, price)
VALUES(11, 102, 1003, '12-Mar-2018', 9.95);
INSERT INTO transactions(transaction_id, patron_id, book_id,
transaction_date, price)
VALUES(12, 103, 1004, '12-Mar-2018', 39.49);
INSERT INTO transactions(transaction_id, patron_id, book_id,
transaction_date, price)
VALUES(13, 103, 1010, '16-Mar-2018', 19.49);
INSERT INTO transactions(transaction_id, patron_id, book_id,
transaction_date, price)
VALUES(14, 105, 1004, '19-Mar-2018', 29.00);
commit;

--drop table transactions;
--drop table patrons;
--drop table books;
```

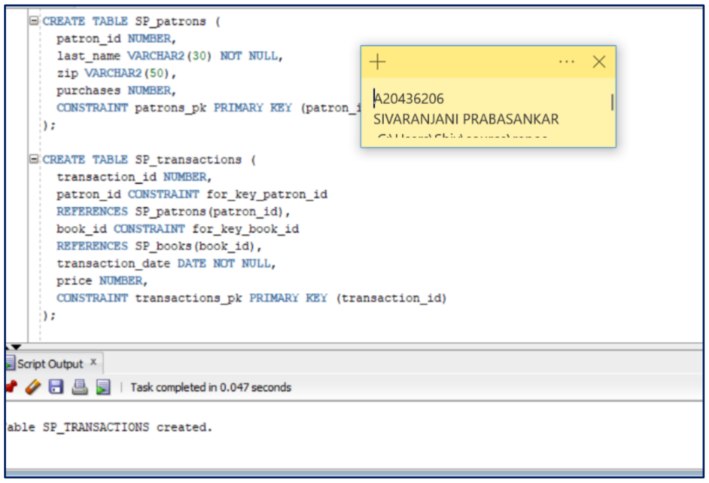


```
CREATE TABLE SP_books
(
  book_id  NUMBER,
  title VARCHAR2(30),
  author_last_name VARCHAR2(20),
  author_first_name VARCHAR2(20),
  classify VARCHAR2(20),
  CONSTRAINT books_pk PRIMARY KEY (book_id)
);

CREATE TABLE SP_patrons (
  patron_id NUMBER,
  last_name VARCHAR2(30) NOT NULL,
  zip VARCHAR2(50),
  purchases NUMBER,
  CONSTRAINT patrons_pk PRIMARY KEY (patron_id)
);

CREATE TABLE SP_transactions (
```

A20436206
SIVARANJANI PRABASANKAR

Task completed in 0.047 seconds

```
Table SP_TRANSACTIONS created.
```

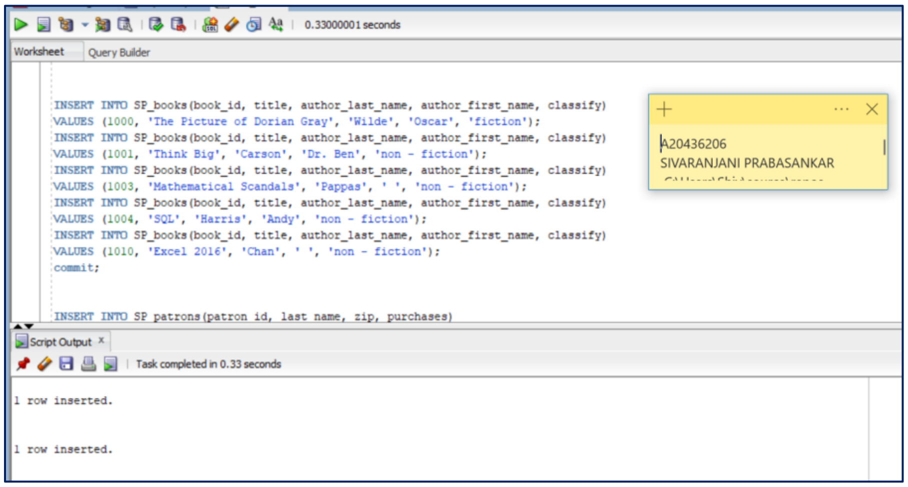


```
CREATE TABLE SP_patrons (
  patron_id NUMBER,
  last_name VARCHAR2(30) NOT NULL,
  zip VARCHAR2(50),
  purchases NUMBER,
  CONSTRAINT patrons_pk PRIMARY KEY (patron_i
);

CREATE TABLE SP_transactions (
  transaction_id NUMBER,
  patron_id CONSTRAINT for_key_patron_id
  REFERENCES SP_patrons(patron_id),
  book_id CONSTRAINT for_key_book_id
  REFERENCES SP_books(book_id),
  transaction_date DATE NOT NULL,
  price NUMBER,
  CONSTRAINT transactions_pk PRIMARY KEY (transaction_id)
);
```

A20436206
SIVARANJANI PRABASANKAR

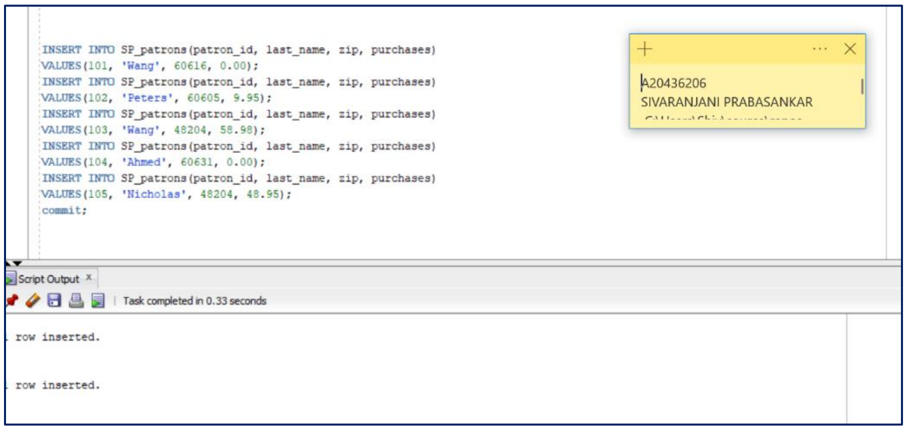
Task completed in 0.047 seconds

```
able SP_TRANSACTIONS created.
```



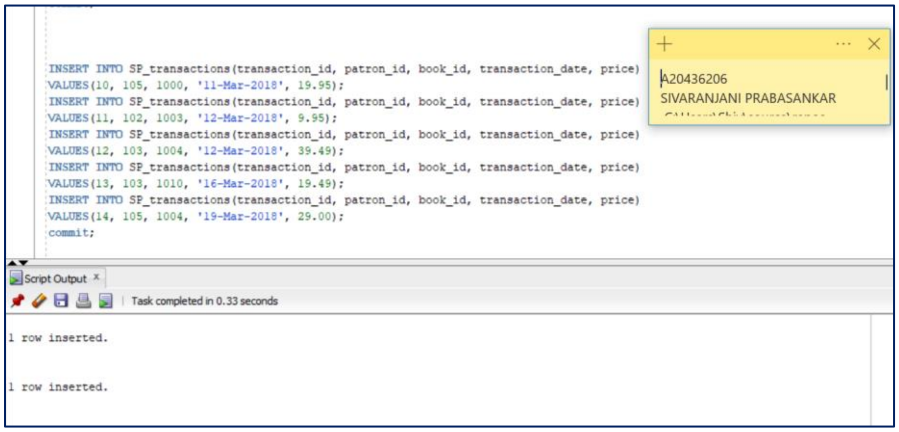
```
INSERT INTO SP_books(book_id, title, author_last_name, author_first_name, classify)
VALUES (1000, 'The Picture of Dorian Gray', 'Wilde', 'Oscar', 'fiction');
INSERT INTO SP_books(book_id, title, author_last_name, author_first_name, classify)
VALUES (1001, 'Think Big', 'Carson', 'Dr. Ben', 'non - fiction');
INSERT INTO SP_books(book_id, title, author_last_name, author_first_name, classify)
VALUES (1003, 'Mathematical Scandals', 'Pappas', ' ', 'non - fiction');
INSERT INTO SP_books(book_id, title, author_last_name, author_first_name, classify)
VALUES (1004, 'SQL', 'Harris', 'Andy', 'non - fiction');
INSERT INTO SP_books(book_id, title, author_last_name, author_first_name, classify)
VALUES (1010, 'Excel 2016', 'Chan', ' ', 'non - fiction');
commit;


INSERT INTO SP_patrons(patron_id, last_name, zip, purchases)
```

```
1 row inserted.


1 row inserted.
```



```
INSERT INTO SP_patrons(patron_id, last_name, zip, purchases)
VALUES(101, 'Wang', 60616, 0.00);
INSERT INTO SP_patrons(patron_id, last_name, zip, purchases)
VALUES(102, 'Peters', 60605, 9.95);
INSERT INTO SP_patrons(patron_id, last_name, zip, purchases)
VALUES(103, 'Wang', 48204, 58.98);
INSERT INTO SP_patrons(patron_id, last_name, zip, purchases)
VALUES(104, 'Ahmed', 60631, 0.00);
INSERT INTO SP_patrons(patron_id, last_name, zip, purchases)
VALUES(105, 'Nicholas', 48204, 48.95);
commit;
```

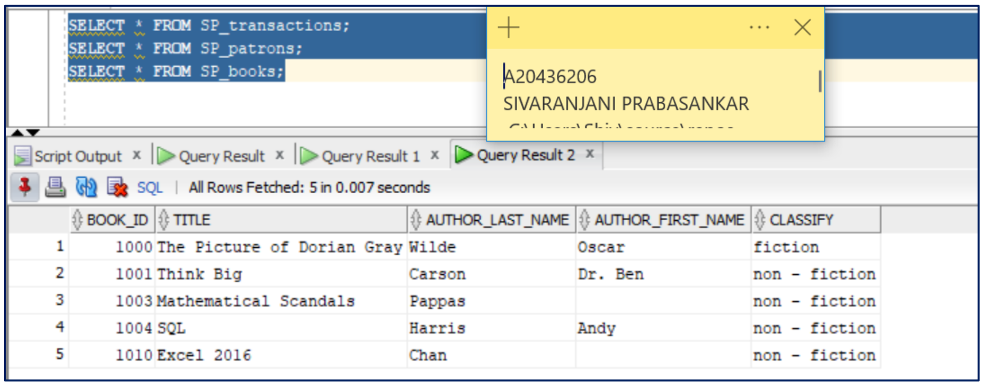
```
 row inserted.


 row inserted.
```



```
INSERT INTO SP_transactions(transaction_id, patron_id, book_id, transaction_date, price)
VALUES(10, 105, 1000, '11-Mar-2018', 19.95);
INSERT INTO SP_transactions(transaction_id, patron_id, book_id, transaction_date, price)
VALUES(11, 102, 1003, '12-Mar-2018', 9.95);
INSERT INTO SP_transactions(transaction_id, patron_id, book_id, transaction_date, price)
VALUES(12, 103, 1004, '12-Mar-2018', 39.49);
INSERT INTO SP_transactions(transaction_id, patron_id, book_id, transaction_date, price)
VALUES(13, 103, 1010, '16-Mar-2018', 19.49);
INSERT INTO SP_transactions(transaction_id, patron_id, book_id, transaction_date, price)
VALUES(14, 105, 1004, '19-Mar-2018', 29.00);
commit;
```

```
1 row inserted.


1 row inserted.
```

Student Name **Sivaranjani Prabasankar** Section ______



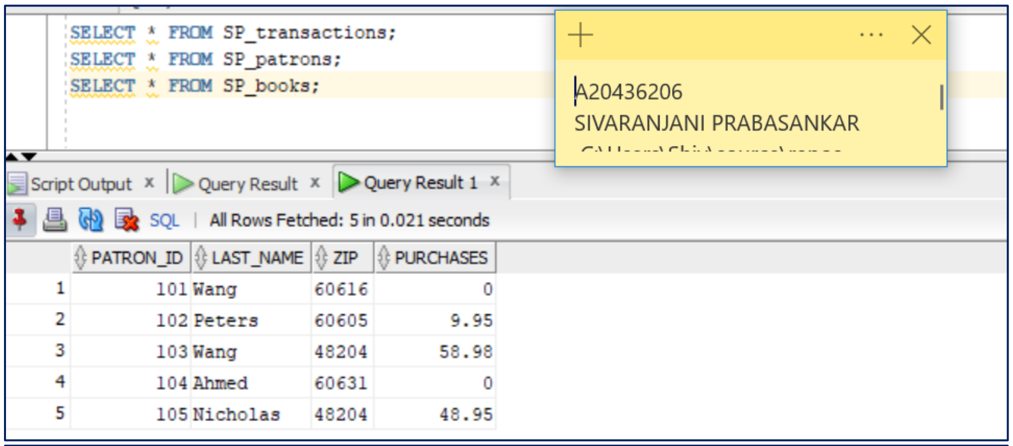
```
SELECT * FROM SP_transactions;
SELECT * FROM SP_patrons;
SELECT * FROM SP_books;
```

A20436206
SIVARANJANI PRABASANKAR

Script Output | Query Result | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 5 in 0.007 seconds

	BOOK_ID	TITLE	AUTHOR_LAST_NAME	AUTHOR_FIRST_NAME	CLASSIFY
1	1000	The Picture of Dorian Gray	Wilde	Oscar	fiction
2	1001	Think Big	Carson	Dr. Ben	non - fiction
3	1003	Mathematical Scandals	Pappas		non - fiction
4	1004	SQL	Harris	Andy	non - fiction
5	1010	Excel 2016	Chan		non - fiction



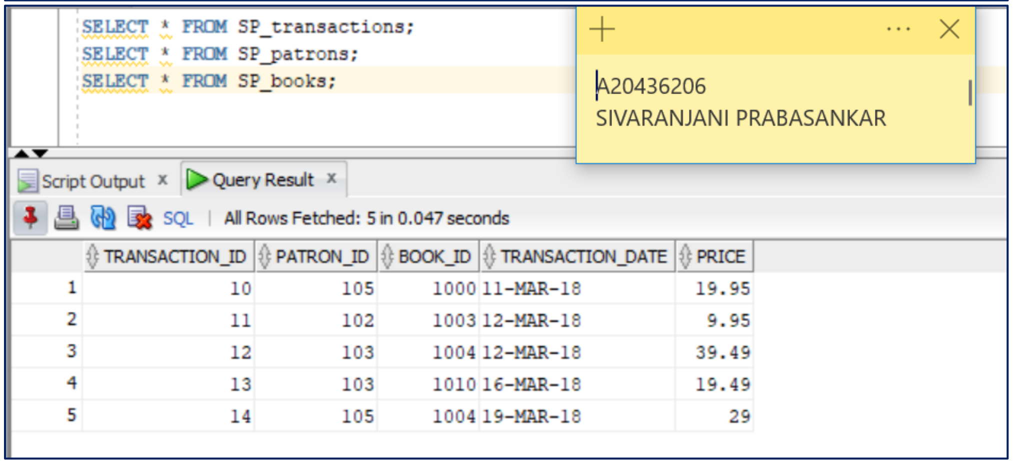
```
SELECT * FROM SP_transactions;
SELECT * FROM SP_patrons;
SELECT * FROM SP_books;
```

A20436206
SIVARANJANI PRABASANKAR

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 5 in 0.021 seconds

	PATRON_ID	LAST_NAME	ZIP	PURCHASES
1	101	Wang	60616	0
2	102	Peters	60605	9.95
3	103	Wang	48204	58.98
4	104	Ahmed	60631	0
5	105	Nicholas	48204	48.95



```
SELECT * FROM SP_transactions;
SELECT * FROM SP_patrons;
SELECT * FROM SP_books;
```

A20436206
SIVARANJANI PRABASANKAR

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.047 seconds

	TRANSACTION_ID	PATRON_ID	BOOK_ID	TRANSACTION_DATE	PRICE
1	10	105	1000	11-MAR-18	19.95
2	11	102	1003	12-MAR-18	9.95
3	12	103	1004	12-MAR-18	39.49
4	13	103	1010	16-MAR-18	19.49
5	14	105	1004	19-MAR-18	29

Part 1 Concepts - Advanced Topics in Data Management

(1) Insert an additional record into each of the three individual data sheets.

For the " books " table, locate the data for a book of your choice from the Barnes & Noble Web site.

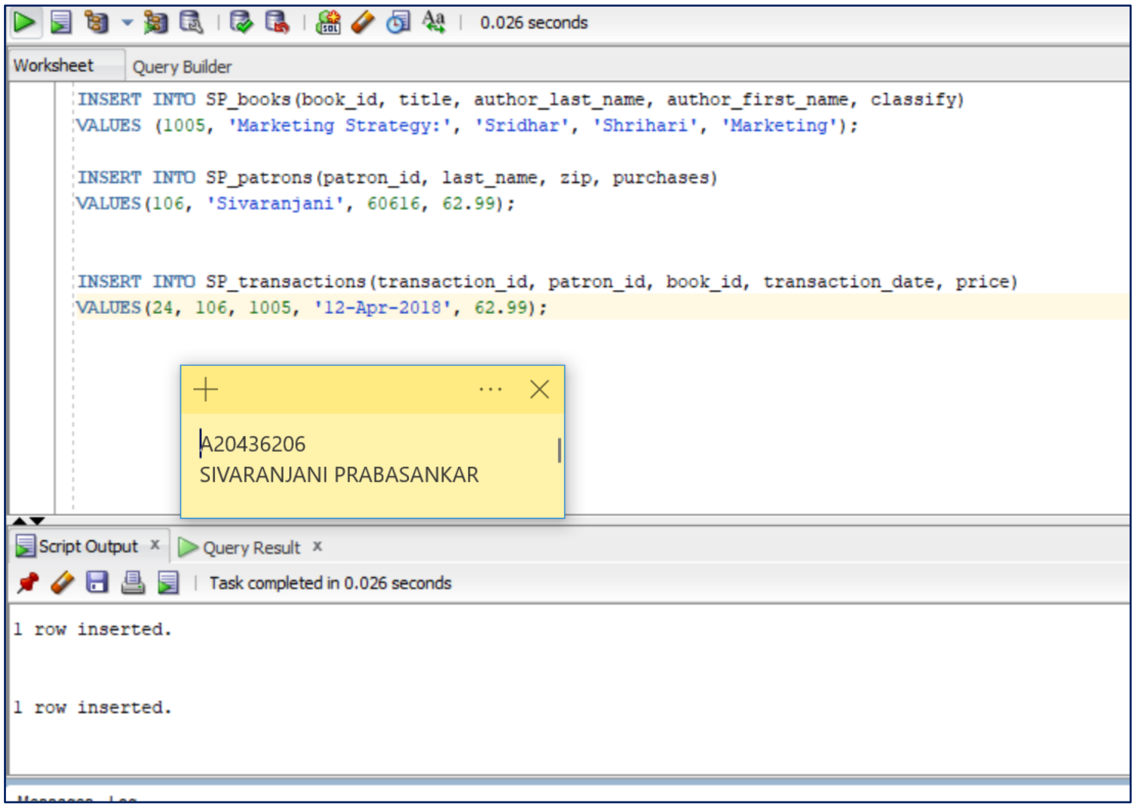
http://www.bn.com

For the " patrons " table, use your own name and postal code.

For the " transactions " table, use the current date for the transaction, yourself for the `patron_id` and the book that you selected as the new record for the " books " table.

Using a `SELECT` statement, display only your newly inserted records from each of the tables.

```
INSERT INTO SP_books(book_id, title, author_last_name, author_first_name,
classify) VALUES (1005, 'Marketing Strategy:', 'Sridhar', 'Shrihari',
'Marketing');

INSERT INTO SP_patrons(patron_id, last_name, zip, purchases) VALUES (106,
'Sivaranjani', 60616, 62.99);

INSERT INTO SP_transactions(transaction_id, patron_id, book_id,
transaction_date, price)VALUES(24, 106, 1005, '12-Apr-2018', 62.99);

SELECT * FROM SP_books B,SP_patrons P, SP_transactions T WHERE P.patron_id =
T.patron_id AND T.book_id = B.book_id AND P.last_name = 'Sivaranjani';
```

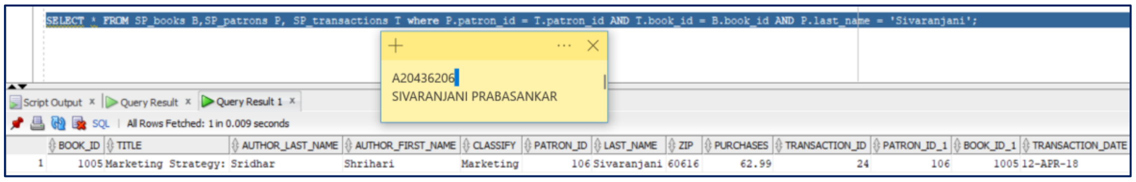


```
INSERT INTO SP_books(book_id, title, author_last_name, author_first_name, classify)
VALUES (1005, 'Marketing Strategy:', 'Sridhar', 'Shrihari', 'Marketing');

INSERT INTO SP_patrons(patron_id, last_name, zip, purchases)
VALUES(106, 'Sivaranjani', 60616, 62.99);


INSERT INTO SP_transactions(transaction_id, patron_id, book_id, transaction_date, price)
VALUES(24, 106, 1005, '12-Apr-2018', 62.99);
```

A20436206
SIVARANJANI PRABASANKAR

Task completed in 0.026 seconds

```
1 row inserted.


1 row inserted.
```



```
SELECT * FROM SP_books B,SP_patrons P, SP_transactions T where P.patron_id = T.patron_id AND T.book_id = B.book_id AND P.last_name = 'Sivaranjani';
```

A20436206
SIVARANJANI PRABASANKAR

All Rows Fetched: 1 in 0.009 seconds

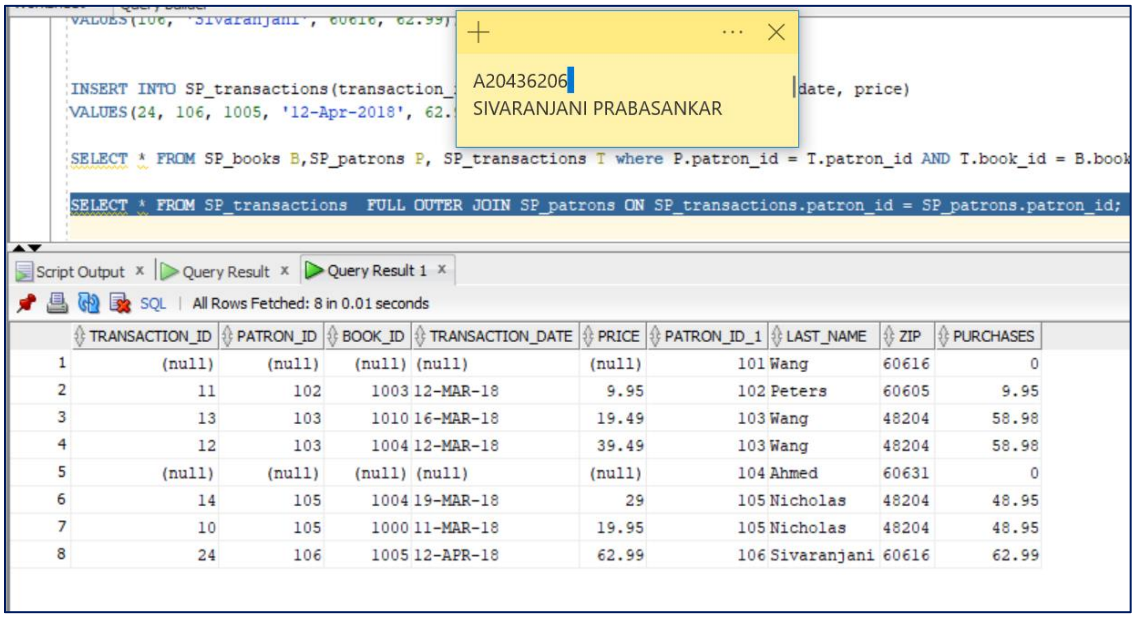
	BOOK_ID	TITLE	AUTHOR_LAST_NAME	AUTHOR_FIRST_NAME	CLASSIFY	PATRON_ID	LAST_NAME	ZIP	PURCHASES	TRANSACTION_ID	PATRON_ID_1	BOOK_ID_1	TRANSACTION_DATE
1	1005	Marketing Strategy:	Sridhar	Shrihari	Marketing	106	Sivaranjani	60616	62.99	24	106	1005	12-APR-18

(2) Perform the following join on your tables.

```
SELECT *
FROM transactions
FULL OUTER JOIN patrons
ON transactions.patron_id = patrons.patron_id;
```

Take a snapshot of the results and paste it below. Also, interpret and discuss the results displayed by this join.

```
SELECT * FROM SP_transactions FULL OUTER JOIN SP_patrons ON
SP_transactions.patron_id = SP_patrons.patron_id;
```



```
INSERT INTO SP_transactions(transaction_                        date, price)
VALUES(24, 106, 1005, '12-Apr-2018', 62.

SELECT * FROM SP_books B,SP_patrons P, SP_transactions T where P.patron_id = T.patron_id AND T.book_id = B.book

SELECT * FROM SP_transactions  FULL OUTER JOIN SP_patrons ON SP_transactions.patron_id = SP_patrons.patron_id;
```

A20436206
SIVARANJANI PRABASANKAR

All Rows Fetched: 8 in 0.01 seconds

	TRANSACTION_ID	PATRON_ID	BOOK_ID	TRANSACTION_DATE	PRICE	PATRON_ID_1	LAST_NAME	ZIP	PURCHASES
1	(null)	(null)	(null)	(null)	(null)	101	Wang	60616	0
2	11	102	1003	12-MAR-18	9.95	102	Peters	60605	9.95
3	13	103	1010	16-MAR-18	19.49	103	Wang	48204	58.98
4	12	103	1004	12-MAR-18	39.49	103	Wang	48204	58.98
5	(null)	(null)	(null)	(null)	(null)	104	Ahmed	60631	0
6	14	105	1004	19-MAR-18	29	105	Nicholas	48204	48.95
7	10	105	1000	11-MAR-18	19.95	105	Nicholas	48204	48.95
8	24	106	1005	12-APR-18	62.99	106	Sivaranjani	60616	62.99

→ The FULL OUTER JOIN keyword returns all matching records from both tables whether the other table matches or not.

→ Here we used OUTER JOIN to combine retrieve all fields in Patron table and with Transaction table and return all records that are common.

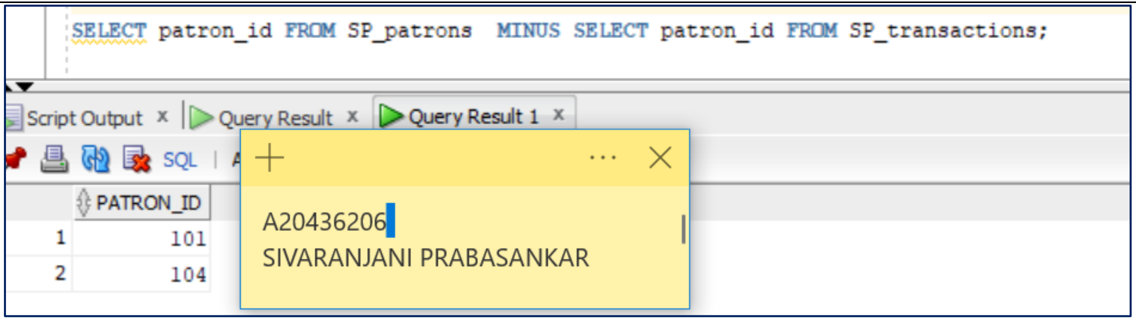
(3) Perform the following set operation on your tables.

Take a snapshot of the results and paste it below. Also, interpret and discuss the results displayed by this set operation.

```
SELECT patron_id
FROM patrons
MINUS
SELECT patron_id
FROM transactions;
```

```
SELECT patron_id FROM SP_patrons  MINUS SELECT patron_id FROM
SP_transactions;
```

Student Name **Sivaranjani Prabasankar** Section



```
SELECT patron_id FROM SP_patrons  MINUS SELECT patron_id FROM SP_transactions;
```

```
   PATRON_ID
1        101
2        104
```

A20436206
SIVARANJANI PRABASANKAR

→ The **SQL MINUS** operator is used to return all rows in the first SELECT statement that are not returned by the second SELECT statement

→ Each SELECT statement will define a dataset.

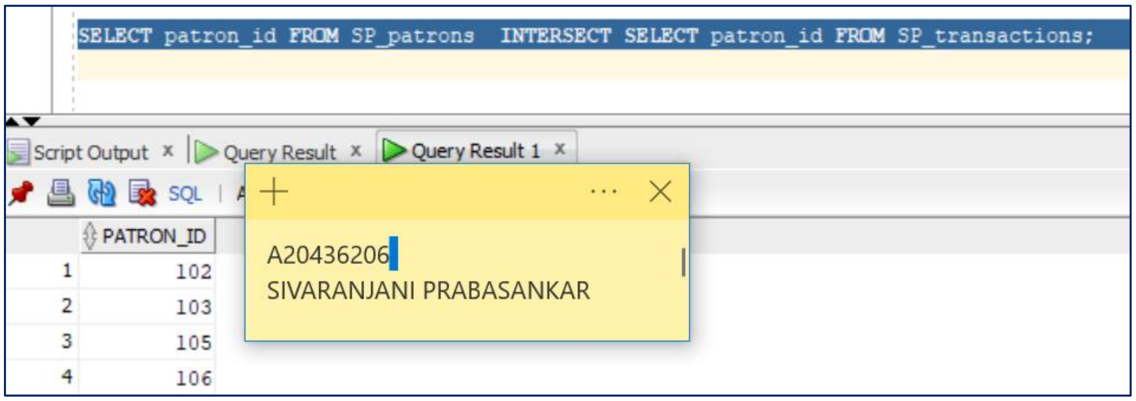
→ Here we used MINUS to combine retrieve all records from in Patron table and from that results remove all records the are common with Transaction table

(4) Perform the following set operation on your tables.

Take a snapshot of the results and paste it below. Also, interpret and discuss the results displayed by this set operation.

```
SELECT patron_id
FROM patrons
INTERSECT
SELECT patron_id
FROM transactions;
```

```
SELECT patron_id FROM SP_patrons  INTERSECT SELECT patron_id FROM
SP_transactions;
```



```
SELECT patron_id FROM SP_patrons  INTERSECT SELECT patron_id FROM SP_transactions;
```

```
   PATRON_ID
1        102
2        103
3        105
4        106
```

A20436206
SIVARANJANI PRABASANKAR

→ The **SQL INTERSECT** operator is used to return the results of 2 or more SELECT statements.

→ It only returns the rows selected by all queries or data sets.

→ If a record exists in one query and not in the other, it will be omitted from the INTERSECT results.

→ Each SELECT statement within the **INTERSECT** must have the same number of fields in the result sets with similar data types.

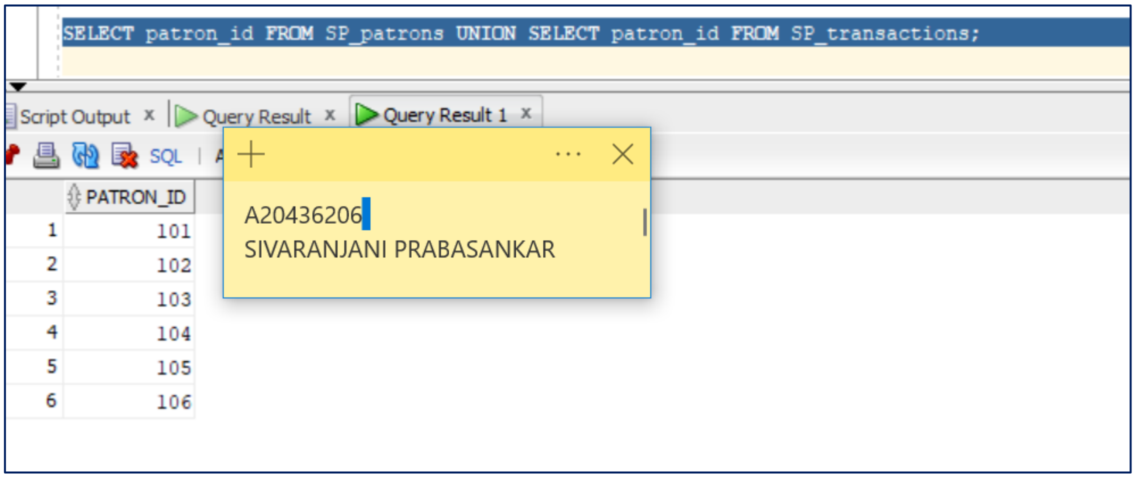
→ Here we used INTERSECT to combine retrieve commons records in Patron table and Transaction table

(5) Perform the following set operation on your tables.

Take a snapshot of the results and paste it below. Also, interpret and discuss the results displayed by this set operation.

```
SELECT patron_id
FROM patrons
UNION
SELECT patron_id
FROM transactions;
```

```
SELECT patron_id FROM SP_patrons UNION SELECT patron_id FROM
SP_transactions;
```



```
SELECT patron_id FROM SP_patrons UNION SELECT patron_id FROM SP_transactions;
```

	PATRON_ID
1	101
2	102
3	103
4	104
5	105
6	106

A20436206
SIVARANJANI PRABASANKAR

→ The **SQL UNION** operator is used to combine the result sets of 2 or more SELECT statements.
→ It removes duplicate rows between the various SELECT statements.
→ Each SELECT statement within the **UNION** must have the same number of fields in the result sets with similar data types.
→ Here we used UNION to retrieve all the records in Patron table and Transaction table without duplications.

Part 2 DBMS Concepts - Advanced Topics in Data Management

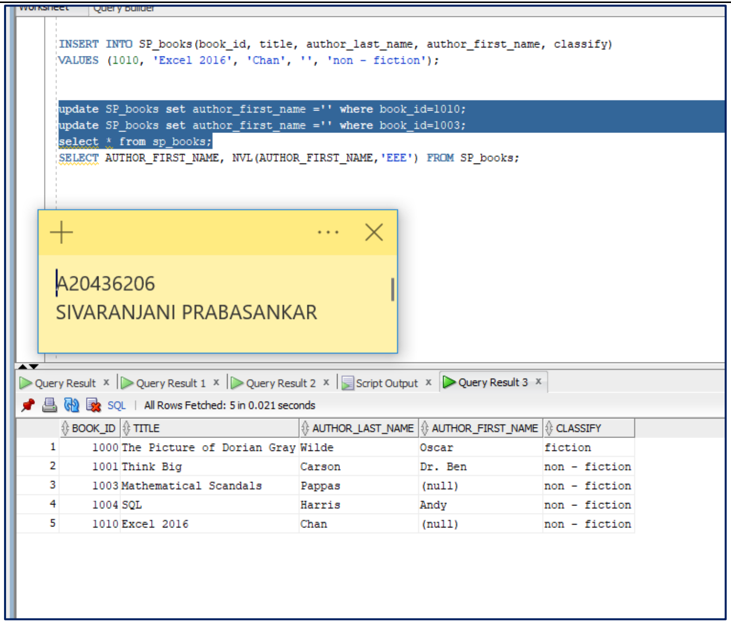
(1) (NULL Values)

Use the Oracle NVL() function on the AUTHOR_FIRST_NAME of the " books " table to replace any empty strings by the NULL value.

The syntax for the NVL function in Oracle and PL / SQL is:

```
NVL( string1, replace_with )
```

```
UPDATE SP_books SET author_first_name ='' WHERE book_id=1010;
UPDATE SP_books SET author_first_name ='' WHERE book_id=1003;
SELECT * FROM sp_books;
SELECT AUTHOR_FIRST_NAME, NVL(AUTHOR_FIRST_NAME,'N/A') FROM SP_books;
```

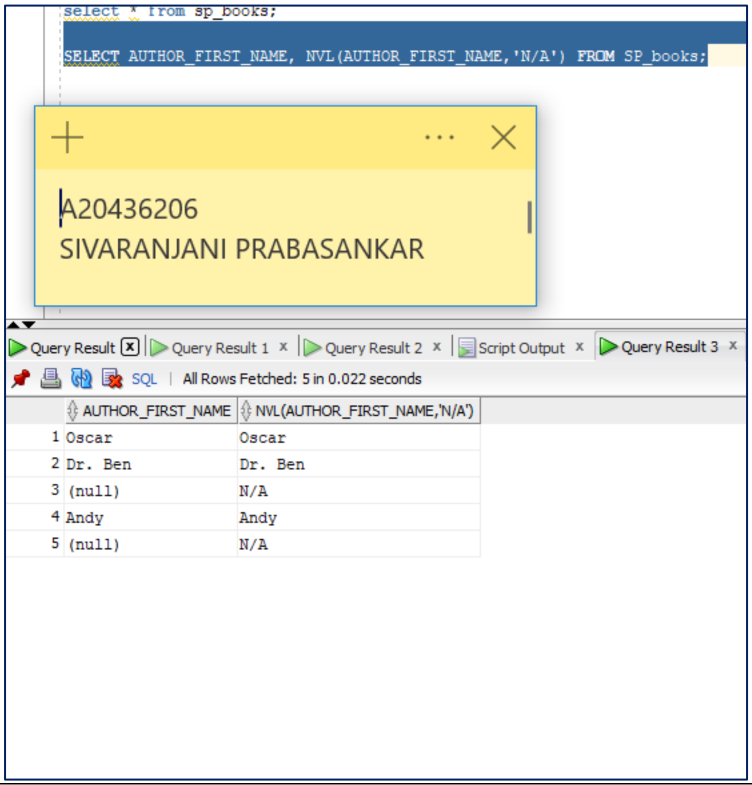


```
INSERT INTO SP_books(book_id, title, author_last_name, author_first_name, classify)
VALUES (1010, 'Excel 2016', 'Chan', '', 'non - fiction');

update SP_books set author_first_name ='' where book_id=1010;
update SP_books set author_first_name ='' where book_id=1003;
select * from sp_books;
SELECT AUTHOR_FIRST_NAME, NVL(AUTHOR_FIRST_NAME,'EEE') FROM SP_books;
```

A20436206
SIVARANJANI PRABASANKAR

All Rows Fetched: 5 in 0.021 seconds

	BOOK_ID	TITLE	AUTHOR_LAST_NAME	AUTHOR_FIRST_NAME	CLASSIFY
1	1000	The Picture of Dorian Gray	Wilde	Oscar	fiction
2	1001	Think Big	Carson	Dr. Ben	non - fiction
3	1003	Mathematical Scandals	Pappas	(null)	non - fiction
4	1004	SQL	Harris	Andy	non - fiction
5	1010	Excel 2016	Chan	(null)	non - fiction



```
select * from sp_books;

SELECT AUTHOR_FIRST_NAME, NVL(AUTHOR_FIRST_NAME,'N/A') FROM SP_books;
```

A20436206
SIVARANJANI PRABASANKAR

All Rows Fetched: 5 in 0.022 seconds

	AUTHOR_FIRST_NAME	NVL(AUTHOR_FIRST_NAME,'N/A')
1	Oscar	Oscar
2	Dr. Ben	Dr. Ben
3	(null)	N/A
4	Andy	Andy
5	(null)	N/A

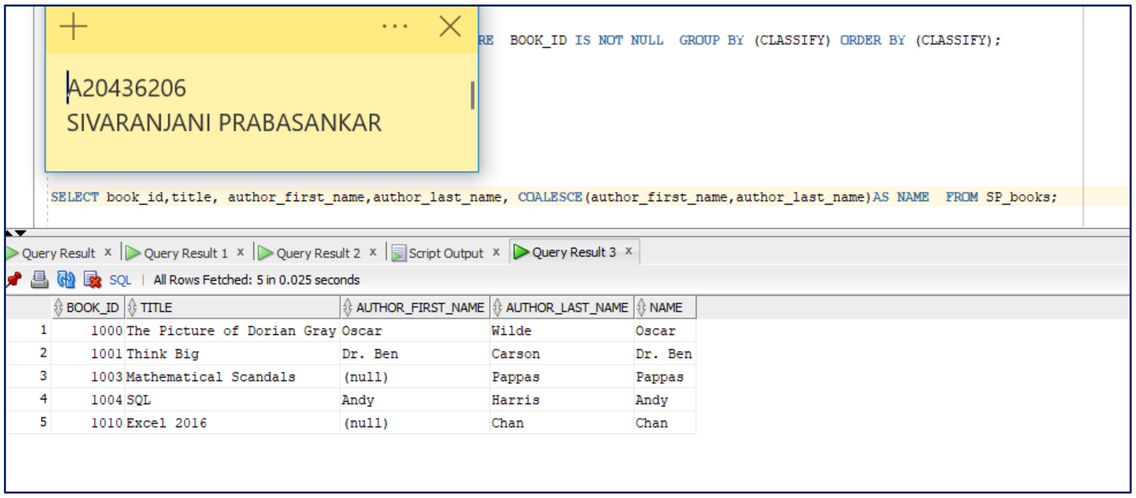
(2) (NULL Values)

Run a query to determine the first non - null expression in the " books " table.

The syntax for the COALESCE function in Oracle and PL / SQL is:

```
COALESCE( expr1, expr2, ... expr_n )
```

```
SELECT book_id,title, author_first_name,author_last_name,
COALESCE(author_first_name,author_last_name)AS NAME  FROM SP_books;
```



A20436206
SIVARANJANI PRABASANKAR

```
RE  BOOK_ID IS NOT NULL  GROUP BY (CLASSIFY) ORDER BY (CLASSIFY);

SELECT book_id,title, author_first_name,author_last_name, COALESCE(author_first_name,author_last_name)AS NAME  FROM SP_books;
```

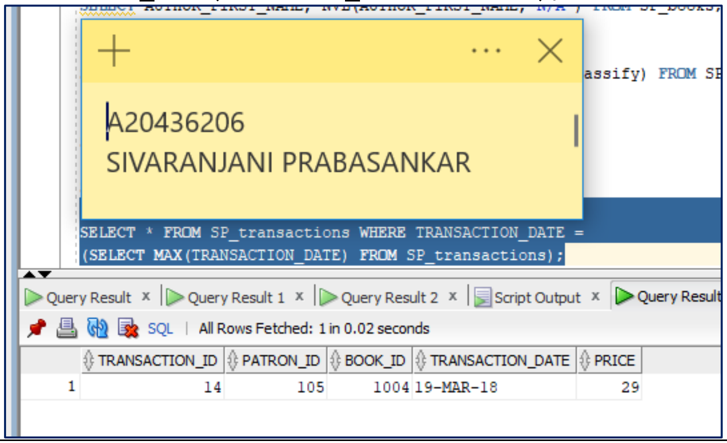
All Rows Fetched: 5 in 0.025 seconds

	BOOK_ID	TITLE	AUTHOR_FIRST_NAME	AUTHOR_LAST_NAME	NAME
1	1000	The Picture of Dorian Gray	Oscar	Wilde	Oscar
2	1001	Think Big	Dr. Ben	Carson	Dr. Ben
3	1003	Mathematical Scandals	(null)	Pappas	Pappas
4	1004	SQL	Andy	Harris	Andy
5	1010	Excel 2016	(null)	Chan	Chan

(3) (Date Arithmetic)

Run a query to determine the most recent transaction from the " transactions " table. Display the date in a Julian format.

```
SELECT * FROM SP_transactions WHERE TRANSACTION_DATE =
(SELECT MAX(TRANSACTION_DATE) FROM SP_transactions);
```



A20436206
SIVARANJANI PRABASANKAR

```
assify) FROM SP

SELECT * FROM SP_transactions WHERE TRANSACTION_DATE =
(SELECT MAX(TRANSACTION_DATE) FROM SP_transactions);
```

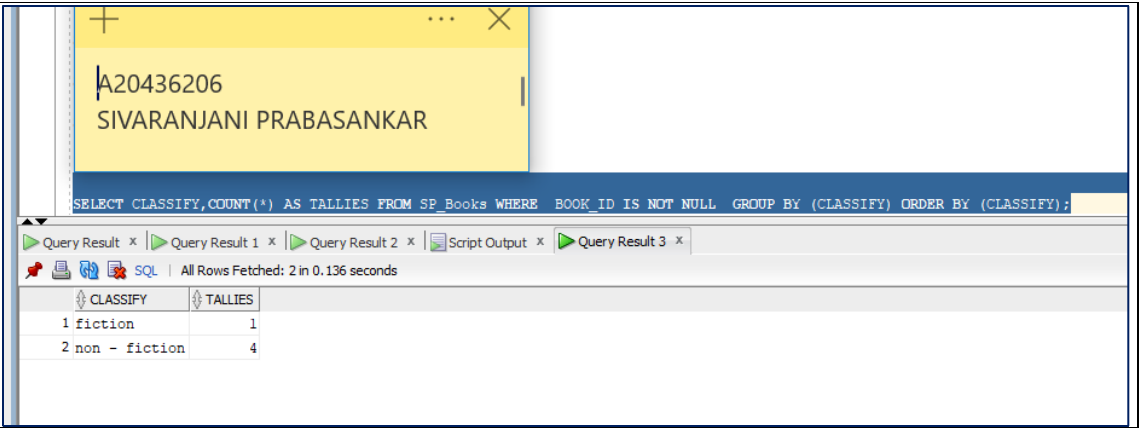
All Rows Fetched: 1 in 0.02 seconds

	TRANSACTION_ID	PATRON_ID	BOOK_ID	TRANSACTION_DATE	PRICE
1	14	105	1004	19-MAR-18	29

(4) (Record Tallies)

Run a query to determine the individual tallies of both the fiction books and the non - fiction books that exist in the " books " table.

```
SELECT CLASSIFY, COUNT (*) AS TALLIES FROM SP_Books WHERE BOOK_ID IS NOT
NULL GROUP BY (CLASSIFY) ORDER BY (CLASSIFY);
```



A20436206
SIVARANJANI PRABASANKAR

```
SELECT CLASSIFY,COUNT(*) AS TALLIES FROM SP_Books WHERE  BOOK_ID IS NOT NULL  GROUP BY (CLASSIFY) ORDER BY (CLASSIFY);
```

All Rows Fetched: 2 in 0.136 seconds

	CLASSIFY	TALLIES
1	fiction	1
2	non - fiction	4

(5) (Data Normalization)

Is the " transactions " table normalized? Explain your answer.

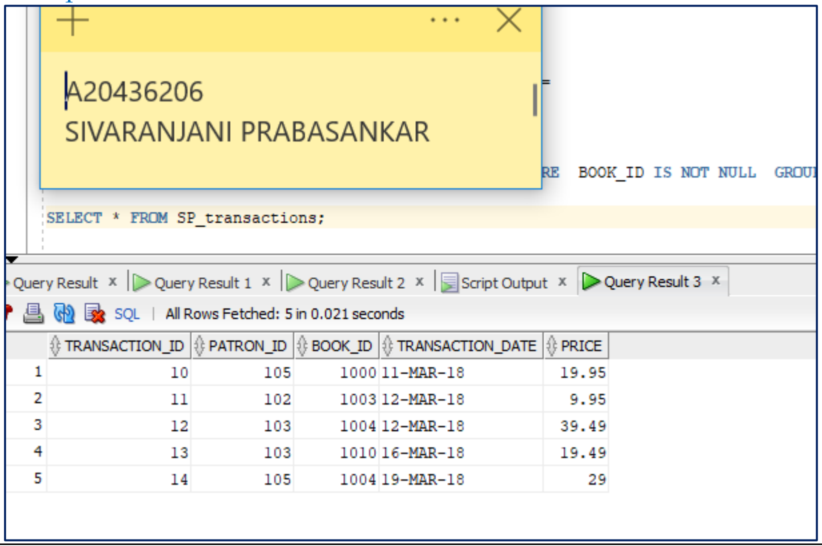
→ The transaction table is passing First Normal Form (1NF) as the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain.

→ The transaction table is passing Second Normal Form (2NF) as we have Single Column Primary Key (Transaction_ID)

→ The transaction table is passing Third Normal Form (3NF) as we have don't have transitive functional dependencies with respect to Patron_ID and Book_ID. 3NF states that all column reference in referenced data that are not dependent on the primary key should be removed. Another way of putting this is that only foreign key columns should be used to reference another table, and no other columns from the parent table should exist in the referenced table.

→ The transaction table is passing Fourth normal form (4NF) where there are no non-trivial multivalued dependencies (i.e. an item depends on more than one value) other than a candidate key.

→ We can't split the table further. Hence the table is normalized form.



A20436206
SIVARANJANI PRABASANKAR

```
SELECT * FROM SP_transactions;
```

All Rows Fetched: 5 in 0.021 seconds

	TRANSACTION_ID	PATRON_ID	BOOK_ID	TRANSACTION_DATE	PRICE
1	10	105	1000	11-MAR-18	19.95
2	11	102	1003	12-MAR-18	9.95
3	12	103	1004	12-MAR-18	39.49
4	13	103	1010	16-MAR-18	19.49
5	14	105	1004	19-MAR-18	29

ITMD 523 *Advanced Topics in Data Management* HW 7

Student Name **Sivaranjani Prabasankar** Section

Part 3 Data Analytics - Advanced Topics in Data Management

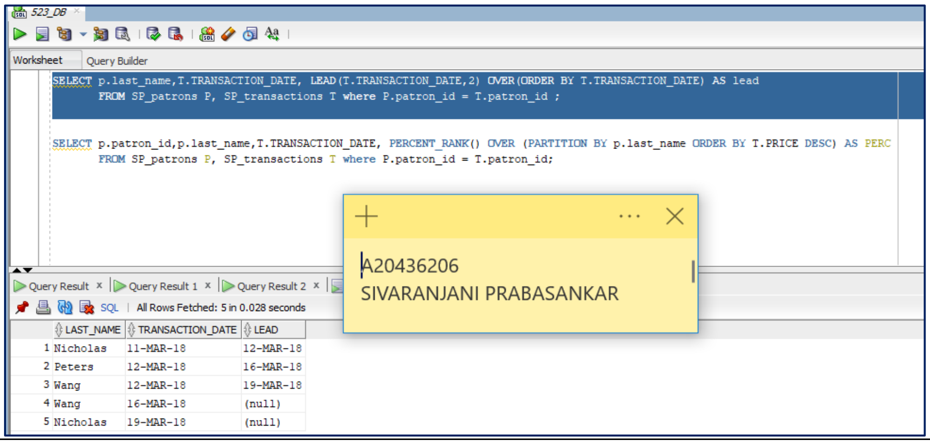
(1) (Data Analytics - Aggregate Functions)

Commonly used for data warehousing environments, analytic functions can compute centered, cumulative, moving and reporting aggregates.

Visit the links below and research the various analytical functions that are available for database use. Examine any examples that are provided at those Web sites.

Specifically for this exercise, delve into the LEAD() analytical function and design an SQL query that uses this function and any or all of your tables [Books, Patrons, Transactions] to return records that are characteristically displayed by such a query. http://psoug.org/reference/analytic_functions.html
http://www.techonthenet.com/oracle/functions/lead.php

```
SELECT p.last_name,T.TRANSACTION_DATE, LEAD(T.TRANSACTION_DATE,2) OVER(ORDER
BY T.TRANSACTION_DATE) AS lead FROM SP_patrons P, SP_transactions T where
P.patron_id = T.patron_id ;
```



```
SELECT p.last_name,T.TRANSACTION_DATE, LEAD(T.TRANSACTION_DATE,2) OVER(ORDER BY T.TRANSACTION_DATE) AS lead
       FROM SP_patrons P, SP_transactions T where P.patron_id = T.patron_id ;

SELECT p.patron_id,p.last_name,T.TRANSACTION_DATE, PERCENT_RANK() OVER (PARTITION BY p.last_name ORDER BY T.PRICE DESC) AS PERC
       FROM SP_patrons P, SP_transactions T where P.patron_id = T.patron_id;
```

A20436206
SIVARANJANI PRABASANKAR

All Rows Fetched: 5 in 0.028 seconds

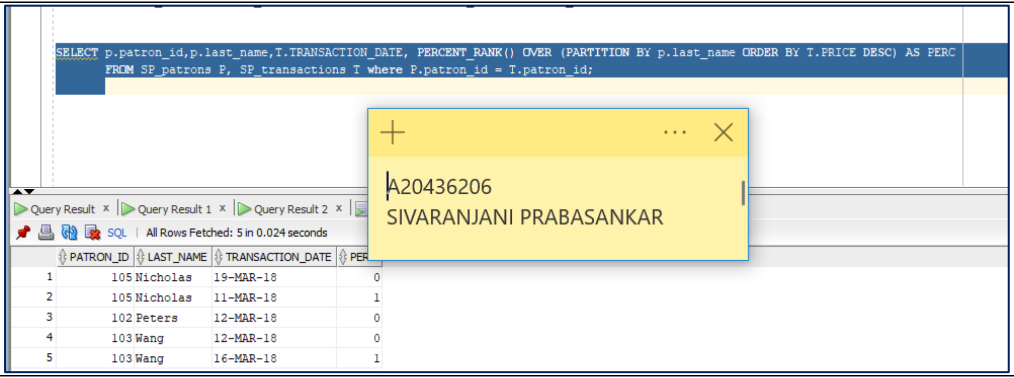
	LAST_NAME	TRANSACTION_DATE	LEAD
1	Nicholas	11-MAR-18	12-MAR-18
2	Peters	12-MAR-18	16-MAR-18
3	Wang	12-MAR-18	19-MAR-18
4	Wang	16-MAR-18	(null)
5	Nicholas	19-MAR-18	(null)

(2) (Data Analytics - Aggregate Functions)

Visit again the Web links that are provided in the preceding exercise.

Specifically for this exercise, investigate the PERCENT_RANK() analytical function and design an SQL query that uses this function and any or all of your tables [Books, Patrons, Transactions] to return records that are characteristically displayed by such a query.

```
SELECT p.patron_id,p.last_name,T.TRANSACTION_DATE, PERCENT_RANK() OVER
(PARTITION BY p.last_name ORDER BY T.PRICE DESC) AS PERC FROM SP_patrons P,
SP_transactions T where P.patron_id = T.patron_id;
```



```
SELECT p.patron_id,p.last_name,T.TRANSACTION_DATE, PERCENT_RANK() OVER (PARTITION BY p.last_name ORDER BY T.PRICE DESC) AS PERC
    FROM SP_patrons P, SP_transactions T where P.patron_id = T.patron_id;
```

All Rows Fetched: 5 in 0.024 seconds

	PATRON_ID	LAST_NAME	TRANSACTION_DATE	PER
1	105	Nicholas	19-MAR-18	0
2	105	Nicholas	11-MAR-18	1
3	102	Peters	12-MAR-18	0
4	103	Wang	12-MAR-18	0
5	103	Wang	16-MAR-18	1

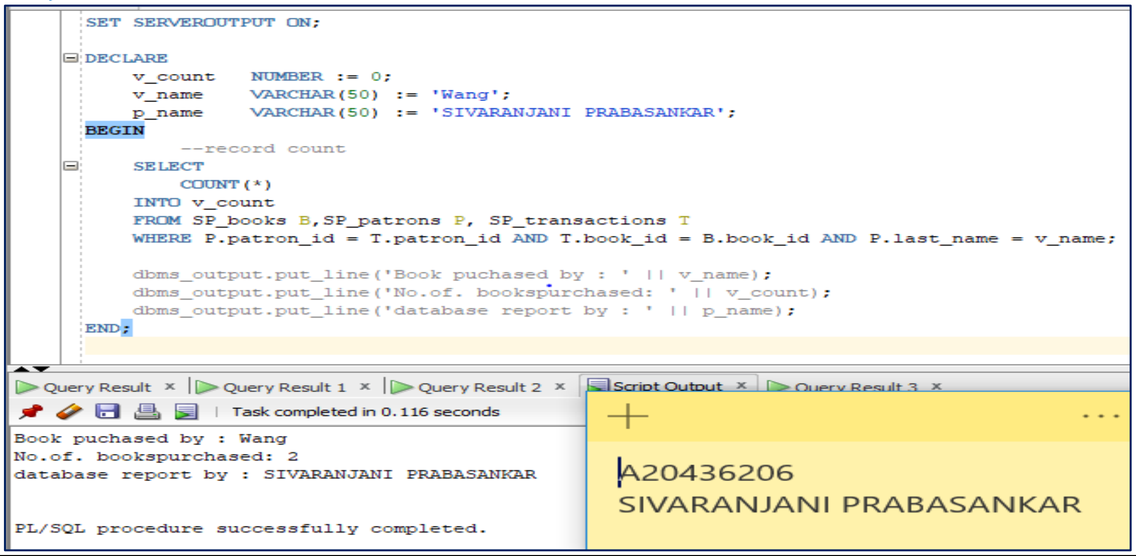
A20436206
SIVARANJANI PRABASANKAR

Part 4 Functions and Procedures - Advanced Topics in Data Management

(1) (Functions and Procedures)

Write a function or procedure in PL / SQL that will determine the number of books that have been purchased by a patron.

```
---script
SET SERVEROUTPUT ON;
DECLARE
    v_count    NUMBER := 0;
    v_name     VARCHAR(50) := 'Wang';
    p_name     VARCHAR(50) := 'SIVARANJANI PRABASANKAR';
BEGIN
SELECT COUNT(*) INTO v_count
 FROM SP_books B,SP_patrons P, SP_transactions T
 WHERE P.patron_id = T.patron_id AND T.book_id = B.book_id AND P.last_name =
v_name;

    dbms_output.put_line('Book puchased by : ' || v_name);
    dbms_output.put_line('No.of. bookspurchased: ' || v_count);
    dbms_output.put_line('database report by : ' || p_name);
END;
```



```
SET SERVEROUTPUT ON;

DECLARE
    v_count    NUMBER := 0;
    v_name     VARCHAR(50) := 'Wang';
    p_name     VARCHAR(50) := 'SIVARANJANI PRABASANKAR';
BEGIN
        --record count
    SELECT
        COUNT(*)
    INTO v_count
    FROM SP_books B,SP_patrons P, SP_transactions T
    WHERE P.patron_id = T.patron_id AND T.book_id = B.book_id AND P.last_name = v_name;

    dbms_output.put_line('Book puchased by : ' || v_name);
    dbms_output.put_line('No.of. bookspurchased: ' || v_count);
    dbms_output.put_line('database report by : ' || p_name);
END;
```

Task completed in 0.116 seconds

```
Book puchased by : Wang
No.of. bookspurchased: 2
database report by : SIVARANJANI PRABASANKAR


PL/SQL procedure successfully completed.
```

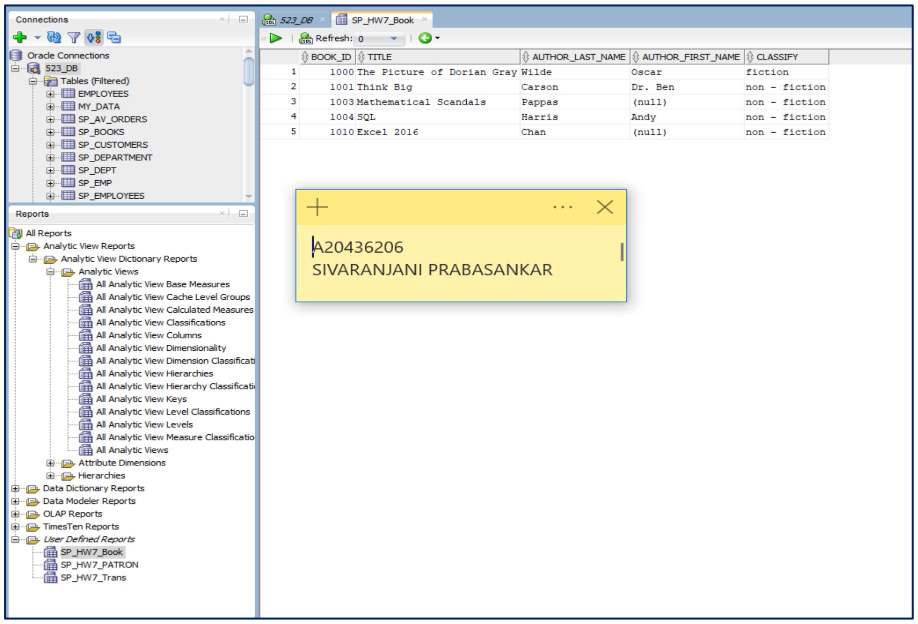
A20436206
SIVARANJANI PRABASANKAR

Student Name **Sivaranjani Prabasankar** Section

(2) (Report Writing)

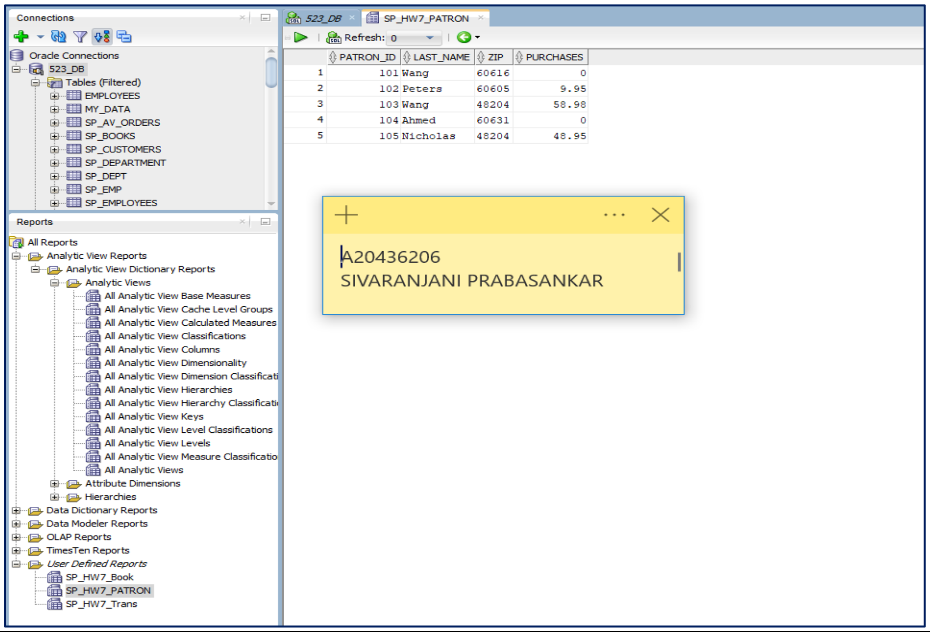
Using any or all of your tables [Books, Patrons, Transactions] to generate a report based on the data.

```
SELECT * FROM SP_transactions;
SELECT * FROM SP_patrons;
SELECT * FROM SP_books;
```



	BOOK_ID	TITLE	AUTHOR_LAST_NAME	AUTHOR_FIRST_NAME	CLASSIFY
1	1000	The Picture of Dorian Gray	Wilde	Oscar	fiction
2	1001	Think Big	Carson	Dr. Ben	non - fiction
3	1003	Mathematical Scandals	Pappas	(null)	non - fiction
4	1004	SQL	Harris	Andy	non - fiction
5	1010	Excel 2016	Chan	(null)	non - fiction

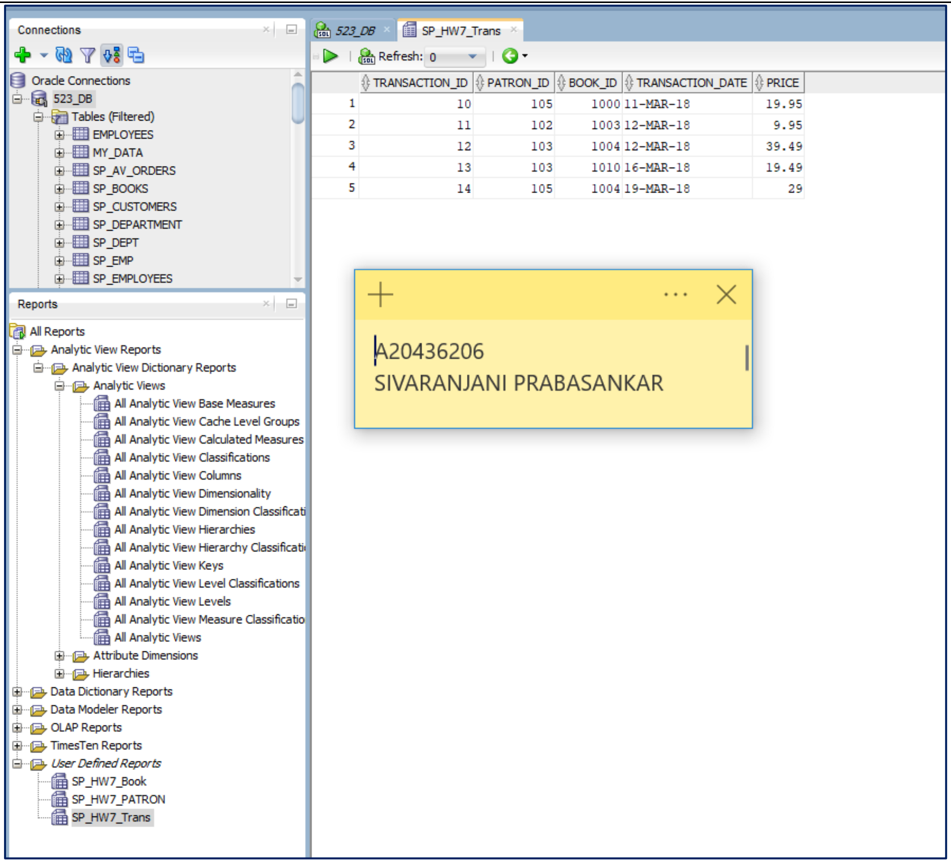
A20436206
SIVARANJANI PRABASANKAR



	PATRON_ID	LAST_NAME	ZIP	PURCHASES
1	101	Wang	60616	0
2	102	Peters	60605	9.95
3	103	Wang	48204	58.98
4	104	Ahmed	60631	0
5	105	Nicholas	48204	48.95

A20436206
SIVARANJANI PRABASANKAR

Student Name **Sivaranjani Prabasankar** Section



	TRANSACTION_ID	PATRON_ID	BOOK_ID	TRANSACTION_DATE	PRICE
1	10	105	1000	11-MAR-18	19.95
2	11	102	1003	12-MAR-18	9.95
3	12	103	1004	12-MAR-18	39.49
4	13	103	1010	16-MAR-18	19.49
5	14	105	1004	19-MAR-18	29

A20436206
SIVARANJANI PRABASANKAR