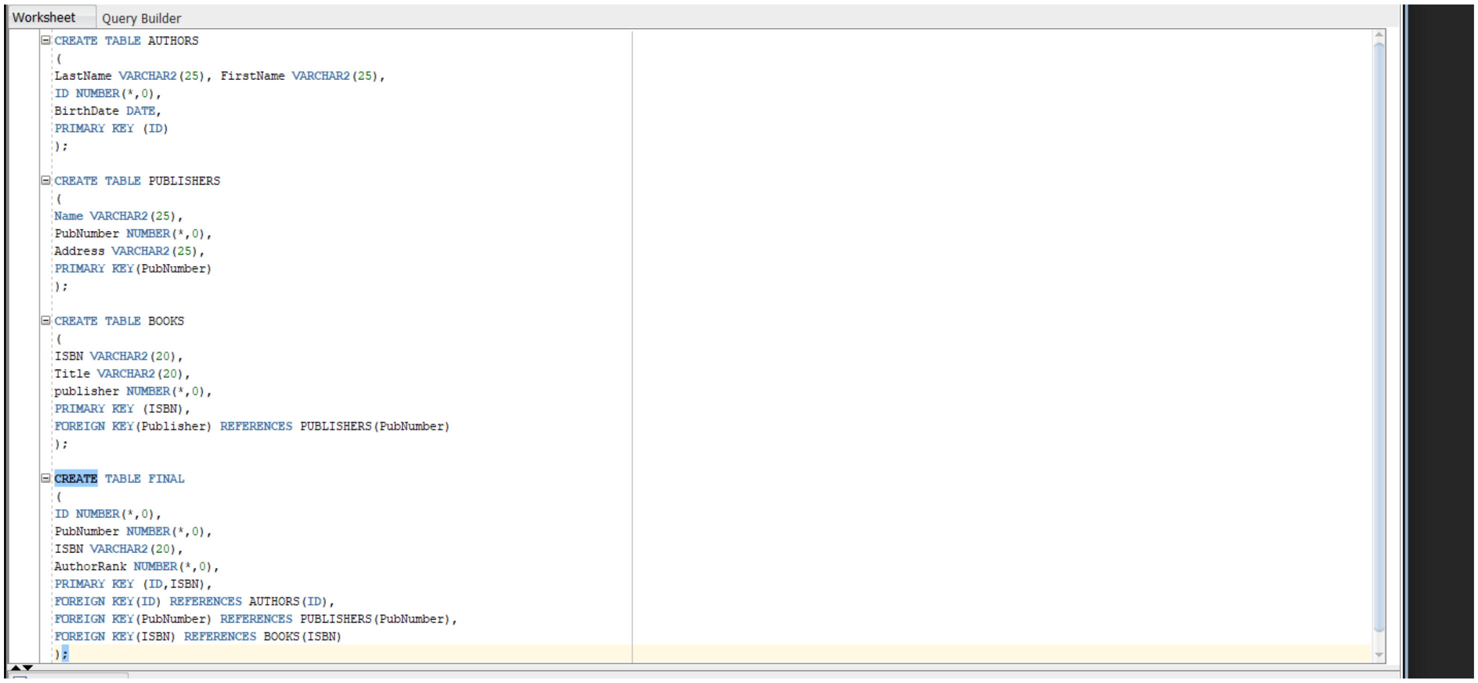
Name: Syed Noor Razi Ali **StudentID : 2070326**

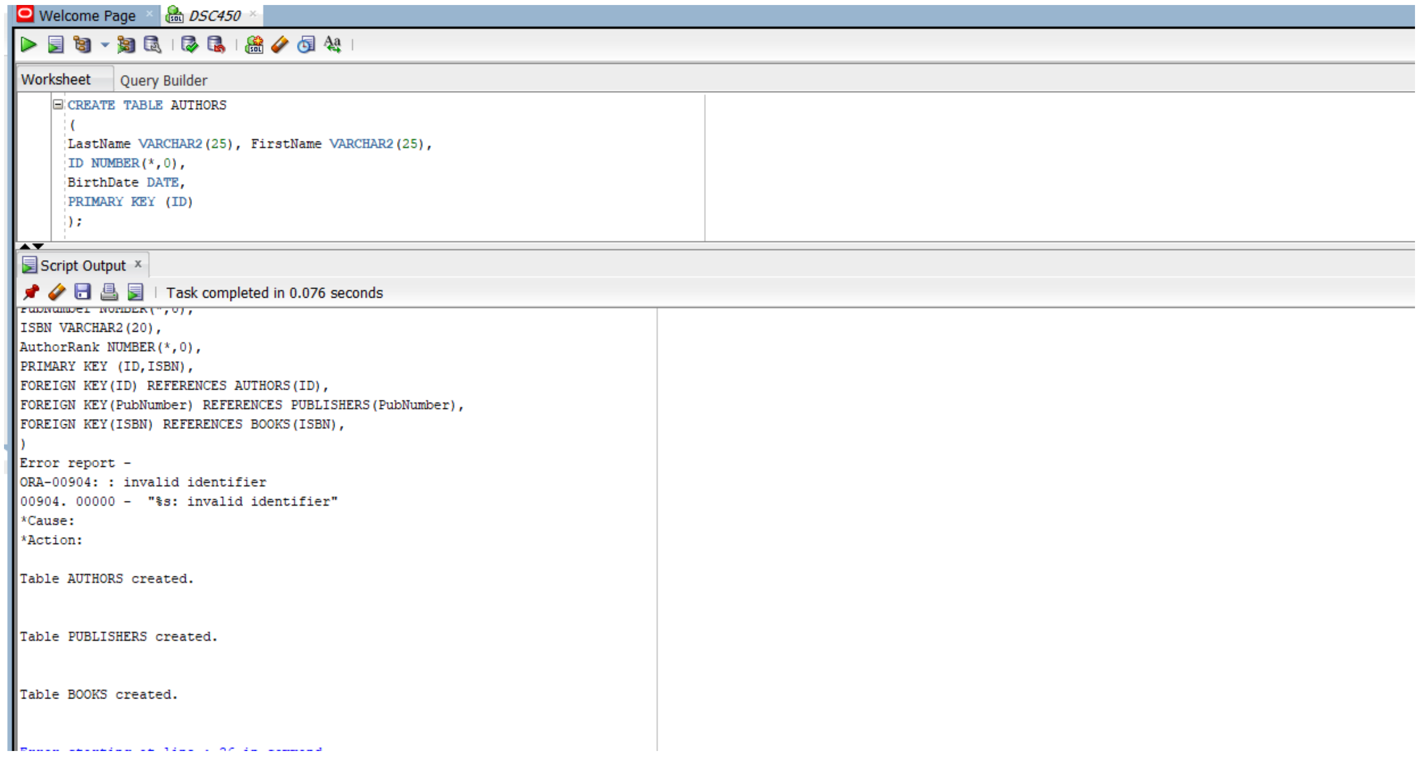
DSC 450: Database Processing for Large-Scale Analytics
Assignment Module 2

Part 1

1) Using your logical schema from previous assignment (Part 2-a), write the necessary SQL DDL script to create the tables. Be sure to specify every primary key and every foreign key. Make <u>reasonable</u> assumptions regarding the attribute domains (for example, setting every column to *VARCHAR2(100)* is not reasonable).



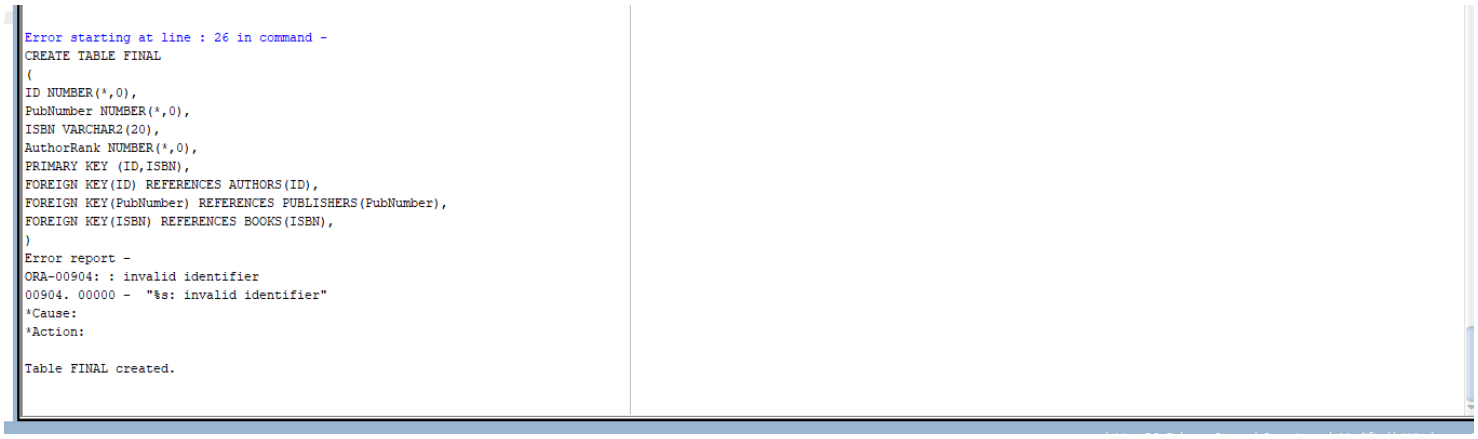
```
CREATE TABLE AUTHORS
(
LastName VARCHAR2(25), FirstName VARCHAR2(25),
ID NUMBER(*,0),
BirthDate DATE,
PRIMARY KEY (ID)
);

CREATE TABLE PUBLISHERS
(
Name VARCHAR2(25),
PubNumber NUMBER(*,0),
Address VARCHAR2(25),
PRIMARY KEY(PubNumber)
);

CREATE TABLE BOOKS
(
ISBN VARCHAR2(20),
Title VARCHAR2(20),
publisher NUMBER(*,0),
PRIMARY KEY (ISBN),
FOREIGN KEY(Publisher) REFERENCES PUBLISHERS(PubNumber)
);

CREATE TABLE FINAL
(
ID NUMBER(*,0),
PubNumber NUMBER(*,0),
ISBN VARCHAR2(20),
AuthorRank NUMBER(*,0),
PRIMARY KEY (ID,ISBN),
FOREIGN KEY(ID) REFERENCES AUTHORS(ID),
FOREIGN KEY(PubNumber) REFERENCES PUBLISHERS(PubNumber),
FOREIGN KEY(ISBN) REFERENCES BOOKS(ISBN)
);
```



```
CREATE TABLE AUTHORS
(
LastName VARCHAR2(25), FirstName VARCHAR2(25),
ID NUMBER(*,0),
BirthDate DATE,
PRIMARY KEY (ID)
);
```

```
PubNumber NUMBER(*,0),
ISBN VARCHAR2(20),
AuthorRank NUMBER(*,0),
PRIMARY KEY (ID,ISBN),
FOREIGN KEY(ID) REFERENCES AUTHORS(ID),
FOREIGN KEY(PubNumber) REFERENCES PUBLISHERS(PubNumber),
FOREIGN KEY(ISBN) REFERENCES BOOKS(ISBN),
)
Error report -
ORA-00904: : invalid identifier
00904. 00000 -  "%s: invalid identifier"
*Cause:
*Action:

Table AUTHORS created.


Table PUBLISHERS created.


Table BOOKS created.


Error starting at line : 26 in command
```

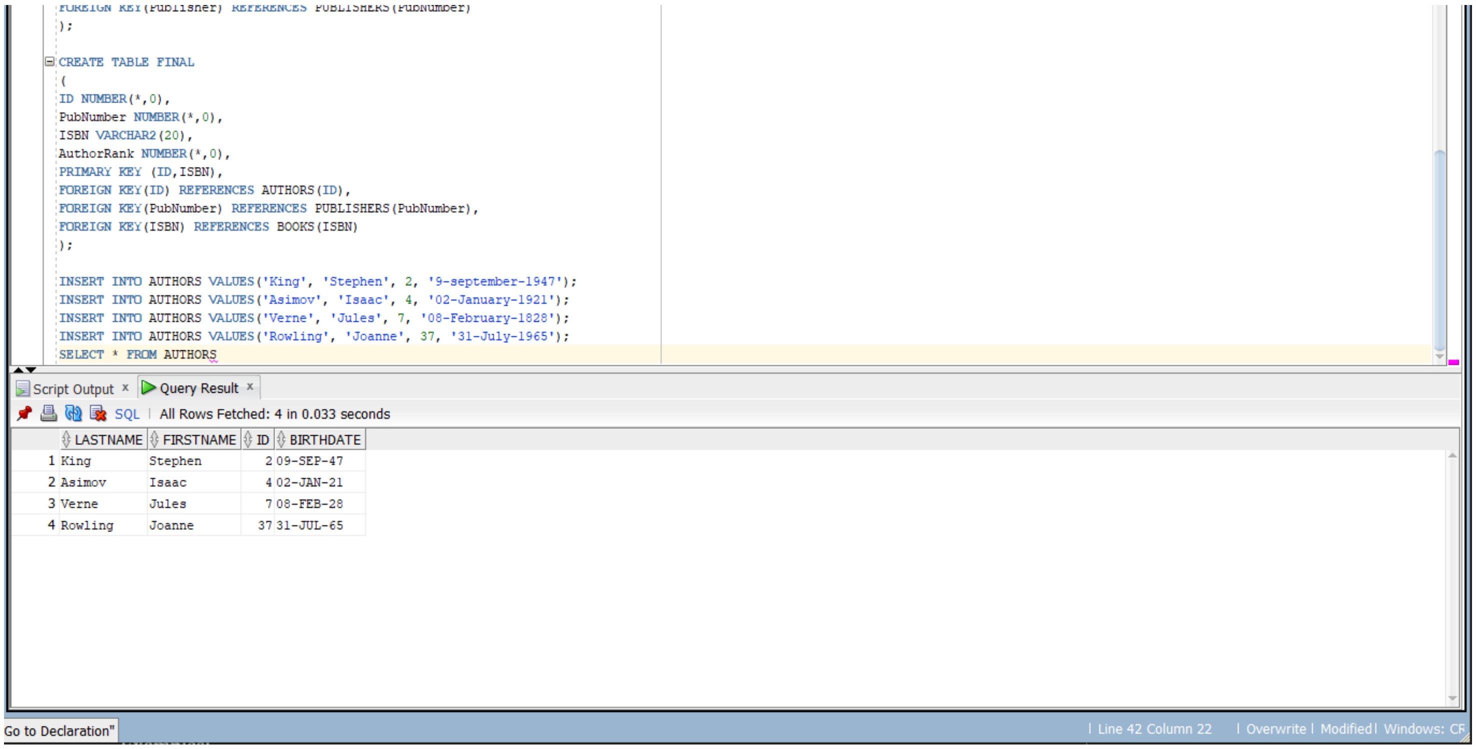
Name: Syed Noor Razi Ali **StudentID : 2070326**



```
Error starting at line : 26 in command -
CREATE TABLE FINAL
(
ID NUMBER(*,0),
PubNumber NUMBER(*,0),
ISBN VARCHAR2(20),
AuthorRank NUMBER(*,0),
PRIMARY KEY (ID,ISBN),
FOREIGN KEY(ID) REFERENCES AUTHORS(ID),
FOREIGN KEY(PubNumber) REFERENCES PUBLISHERS(PubNumber),
FOREIGN KEY(ISBN) REFERENCES BOOKS(ISBN),
)
Error report -
ORA-00904: : invalid identifier
00904. 00000 -  "%s: invalid identifier"
*Cause:
*Action:

Table FINAL created.
```

2) Write SQL INSERT statements to populate your database from Part 2-a with the following data (NOTE: remember that strings would need to use single quotes, e.g., 'Asimov'). Be sure to verify that your statements worked correctly and loaded the data in Oracle (see tutorial on how to connect to Oracle using SQLDeveloper).

a) (King, Stephen, 2, September-9-1947)
b) (Asimov, Isaac, 4, January 2 1921)
c) (Verne, Jules, 7, February 8 1828)
d) (Rowling, Joanne, 37, July 31 1965)

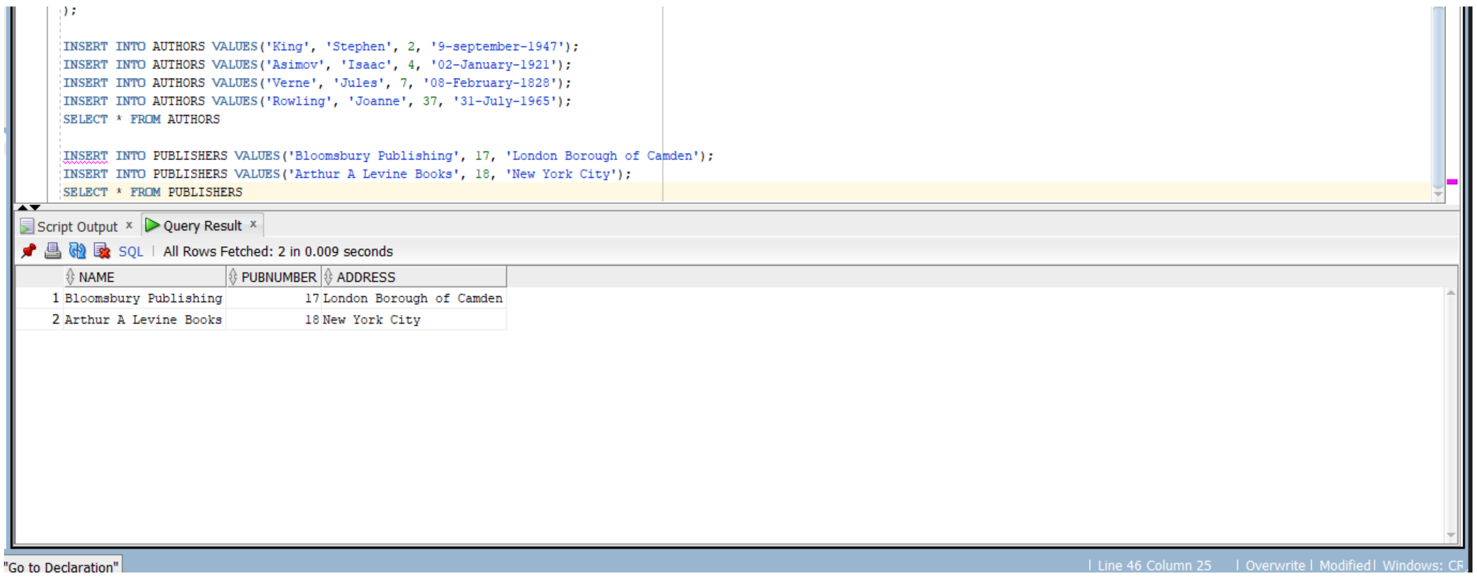


```
FOREIGN KEY(Publisher) REFERENCES PUBLISHERS(PubNumber)
);

CREATE TABLE FINAL
(
ID NUMBER(*,0),
PubNumber NUMBER(*,0),
ISBN VARCHAR2(20),
AuthorRank NUMBER(*,0),
PRIMARY KEY (ID,ISBN),
FOREIGN KEY(ID) REFERENCES AUTHORS(ID),
FOREIGN KEY(PubNumber) REFERENCES PUBLISHERS(PubNumber),
FOREIGN KEY(ISBN) REFERENCES BOOKS(ISBN)
);

INSERT INTO AUTHORS VALUES('King', 'Stephen', 2, '9-september-1947');
INSERT INTO AUTHORS VALUES('Asimov', 'Isaac', 4, '02-January-1921');
INSERT INTO AUTHORS VALUES('Verne', 'Jules', 7, '08-February-1828');
INSERT INTO AUTHORS VALUES('Rowling', 'Joanne', 37, '31-July-1965');
SELECT * FROM AUTHORS
```

All Rows Fetched: 4 in 0.033 seconds

```
  LASTNAME  FIRSTNAME  ID  BIRTHDATE
1 King      Stephen     2  09-SEP-47
2 Asimov    Isaac       4  02-JAN-21
3 Verne     Jules       7  08-FEB-28
4 Rowling   Joanne     37  31-JUL-65
```

e) (Bloomsbury Publishing, 17, London Borough of Camden)
f) (Arthur A Levine Books, 18, New York City)

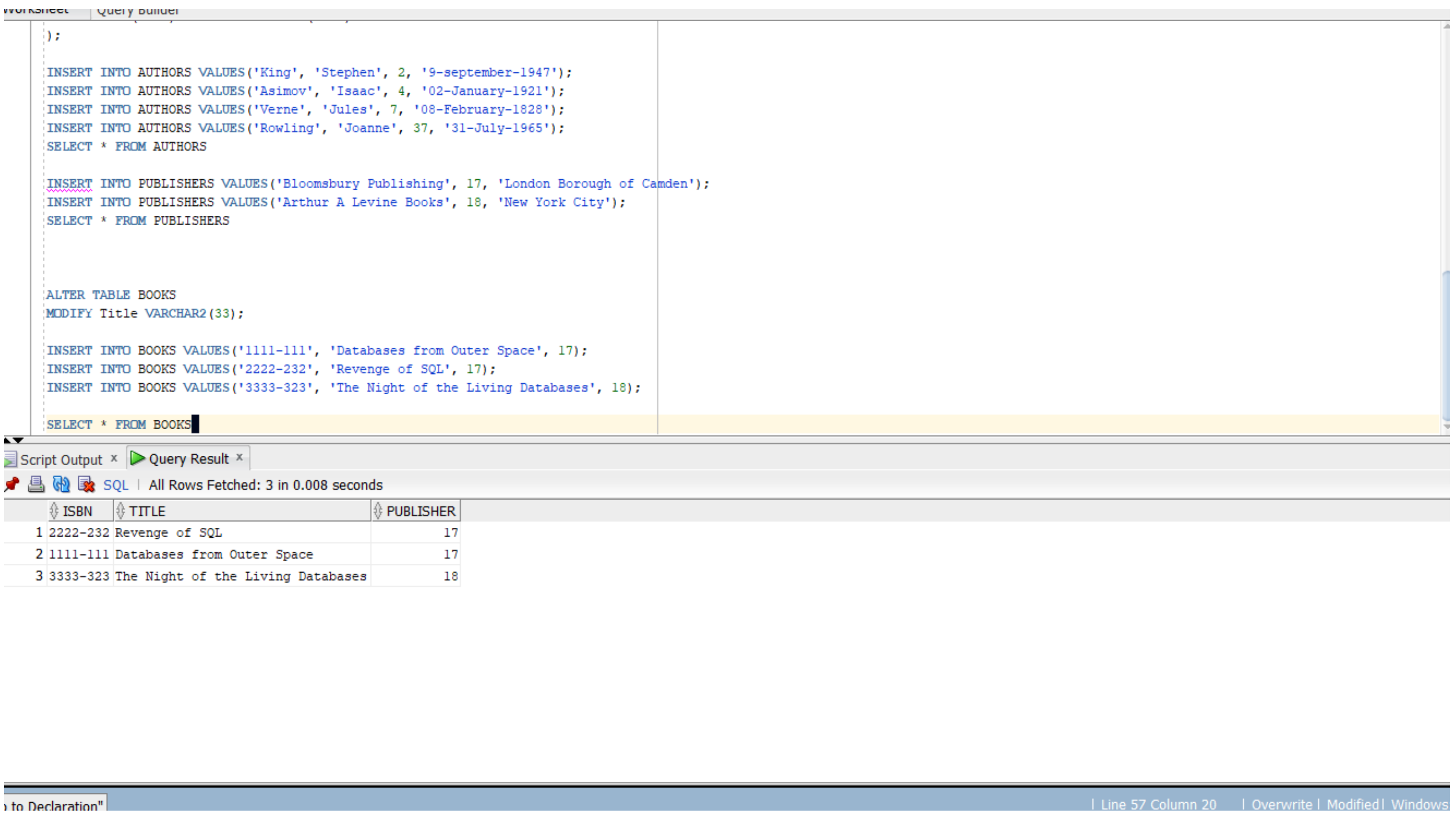


```
);

INSERT INTO AUTHORS VALUES('King', 'Stephen', 2, '9-september-1947');
INSERT INTO AUTHORS VALUES('Asimov', 'Isaac', 4, '02-January-1921');
INSERT INTO AUTHORS VALUES('Verne', 'Jules', 7, '08-February-1828');
INSERT INTO AUTHORS VALUES('Rowling', 'Joanne', 37, '31-July-1965');
SELECT * FROM AUTHORS

INSERT INTO PUBLISHERS VALUES('Bloomsbury Publishing', 17, 'London Borough of Camden');
INSERT INTO PUBLISHERS VALUES('Arthur A Levine Books', 18, 'New York City');
SELECT * FROM PUBLISHERS
```

All Rows Fetched: 2 in 0.009 seconds

```
  NAME                   PUBNUMBER ADDRESS
1 Bloomsbury Publishing         17 London Borough of Camden
2 Arthur A Levine Books         18 New York City
```

g) (1111-111, Databases from Outer Space, 17)
h) (2222-232, Revenge of SQL, 17)
i) (3333-323, The Night of the Living Databases, 18)



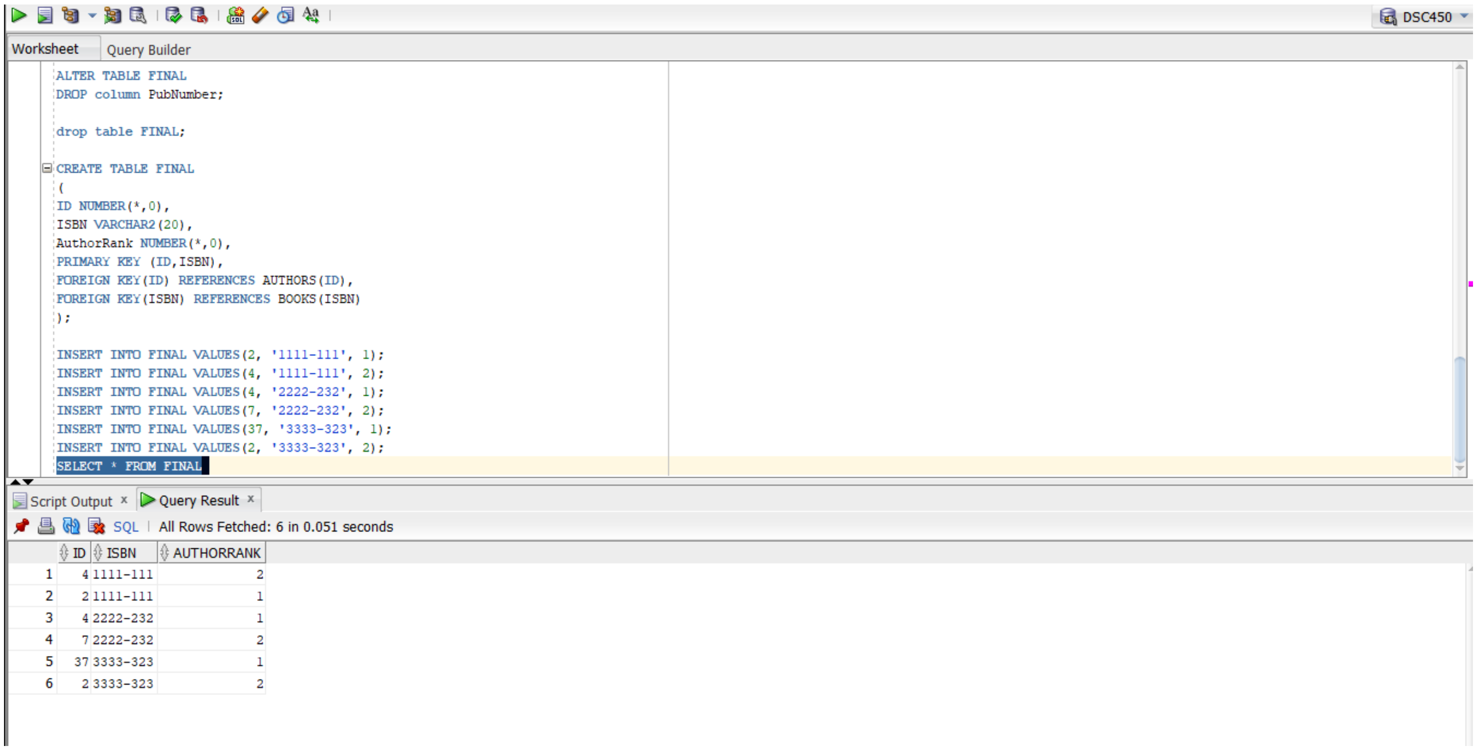
```
);

INSERT INTO AUTHORS VALUES('King', 'Stephen', 2, '9-september-1947');
INSERT INTO AUTHORS VALUES('Asimov', 'Isaac', 4, '02-January-1921');
INSERT INTO AUTHORS VALUES('Verne', 'Jules', 7, '08-February-1828');
INSERT INTO AUTHORS VALUES('Rowling', 'Joanne', 37, '31-July-1965');
SELECT * FROM AUTHORS

INSERT INTO PUBLISHERS VALUES('Bloomsbury Publishing', 17, 'London Borough of Camden');
INSERT INTO PUBLISHERS VALUES('Arthur A Levine Books', 18, 'New York City');
SELECT * FROM PUBLISHERS



ALTER TABLE BOOKS
MODIFY Title VARCHAR2(33);

INSERT INTO BOOKS VALUES('1111-111', 'Databases from Outer Space', 17);
INSERT INTO BOOKS VALUES('2222-232', 'Revenge of SQL', 17);
INSERT INTO BOOKS VALUES('3333-323', 'The Night of the Living Databases', 18);

SELECT * FROM BOOKS
```

All Rows Fetched: 3 in 0.008 seconds

```
  ISBN     TITLE                             PUBLISHER
1 2222-232 Revenge of SQL                           17
2 1111-111 Databases from Outer Space               17
3 3333-323 The Night of the Living Databases        18
```

j) (2, 1111-111, 1)
k) (4, 1111-111, 2)
l) (4, 2222-232, 1)
m) (7, 2222-232, 2)
n) (37, 3333-323, 1)
o) (2, 3333-323, 2)

Name: Syed Noor Razi Ali **StudentID : 2070326**

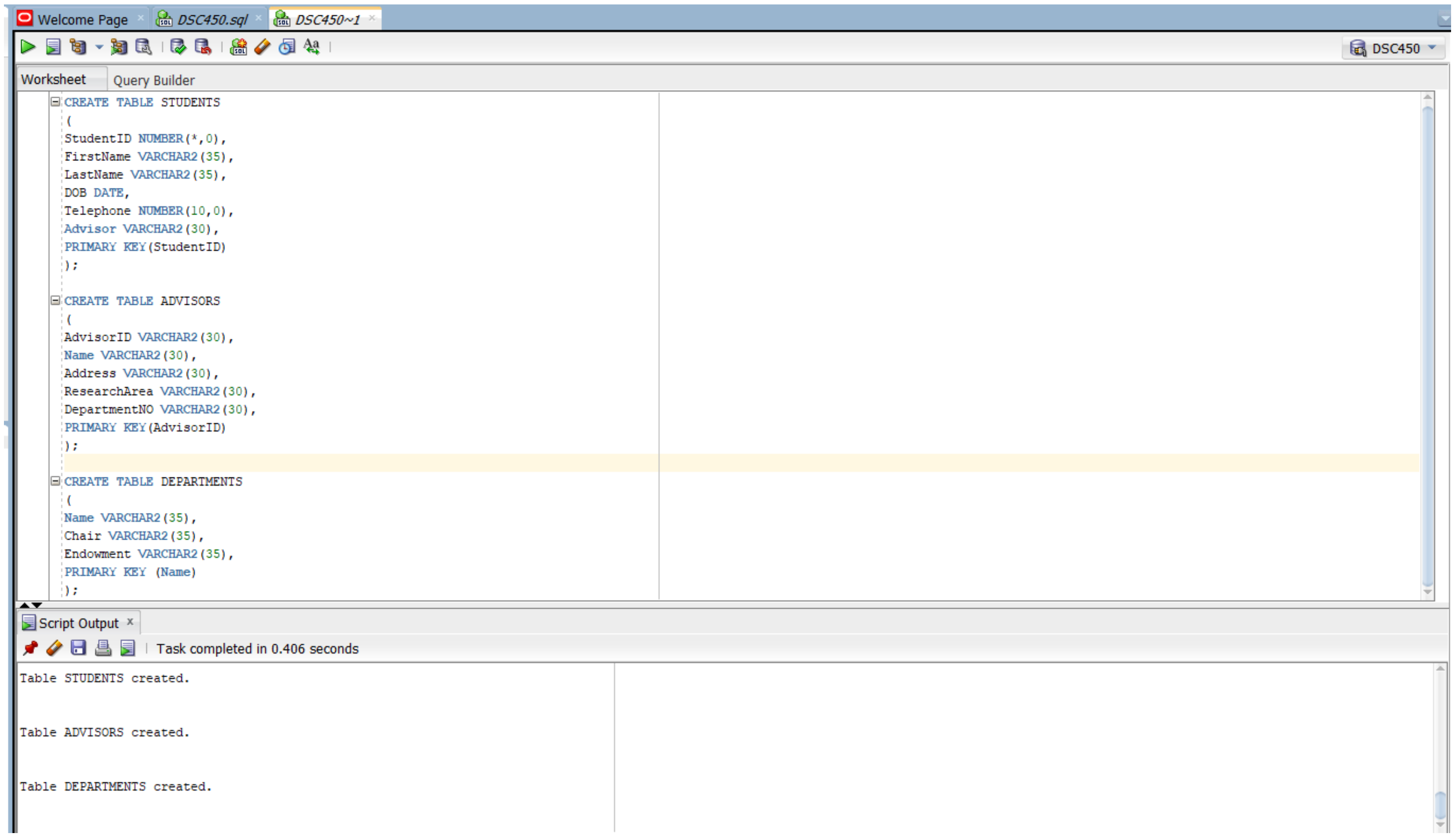


```
ALTER TABLE FINAL
DROP column PubNumber;

drop table FINAL;

CREATE TABLE FINAL
(
ID NUMBER(*,0),
ISBN VARCHAR2(20),
AuthorRank NUMBER(*,0),
PRIMARY KEY (ID,ISBN),
FOREIGN KEY(ID) REFERENCES AUTHORS(ID),
FOREIGN KEY(ISBN) REFERENCES BOOKS(ISBN)
);

INSERT INTO FINAL VALUES(2, '1111-111', 1);
INSERT INTO FINAL VALUES(4, '1111-111', 2);
INSERT INTO FINAL VALUES(4, '2222-232', 1);
INSERT INTO FINAL VALUES(7, '2222-232', 2);
INSERT INTO FINAL VALUES(37, '3333-323', 1);
INSERT INTO FINAL VALUES(2, '3333-323', 2);
SELECT * FROM FINAL
```

All Rows Fetched: 6 in 0.051 seconds

	ID	ISBN	AUTHORRANK
1	4	1111-111	2
2	2	1111-111	1
3	4	2222-232	1
4	7	2222-232	2
5	37	3333-323	1
6	2	3333-323	2

3) **Using logical schema from previous assignment (Part 2-b) write the necessary SQL DDL script to create the tables. Be sure to specify every primary key and every foreign key.**



```
CREATE TABLE STUDENTS
(
StudentID NUMBER(*,0),
FirstName VARCHAR2(35),
LastName VARCHAR2(35),
DOB DATE,
Telephone NUMBER(10,0),
Advisor VARCHAR2(30),
PRIMARY KEY(StudentID)
);

CREATE TABLE ADVISORS
(
AdvisorID VARCHAR2(30),
Name VARCHAR2(30),
Address VARCHAR2(30),
ResearchArea VARCHAR2(30),
DepartmentNO VARCHAR2(30),
PRIMARY KEY(AdvisorID)
);

CREATE TABLE DEPARTMENTS
(
Name VARCHAR2(35),
Chair VARCHAR2(35),
Endowment VARCHAR2(35),
PRIMARY KEY (Name)
);
```

Task completed in 0.406 seconds

```
Table STUDENTS created.


Table ADVISORS created.


Table DEPARTMENTS created.
```

4) **Write a python function to validate SQL insert statements. The function will take a string containing a SQL insert statement and print two kinds of messages. 1) “Invalid insert” or 2) Inserting [list of values] into [the target] table. The values and the table name would be based on each particular statement. For**

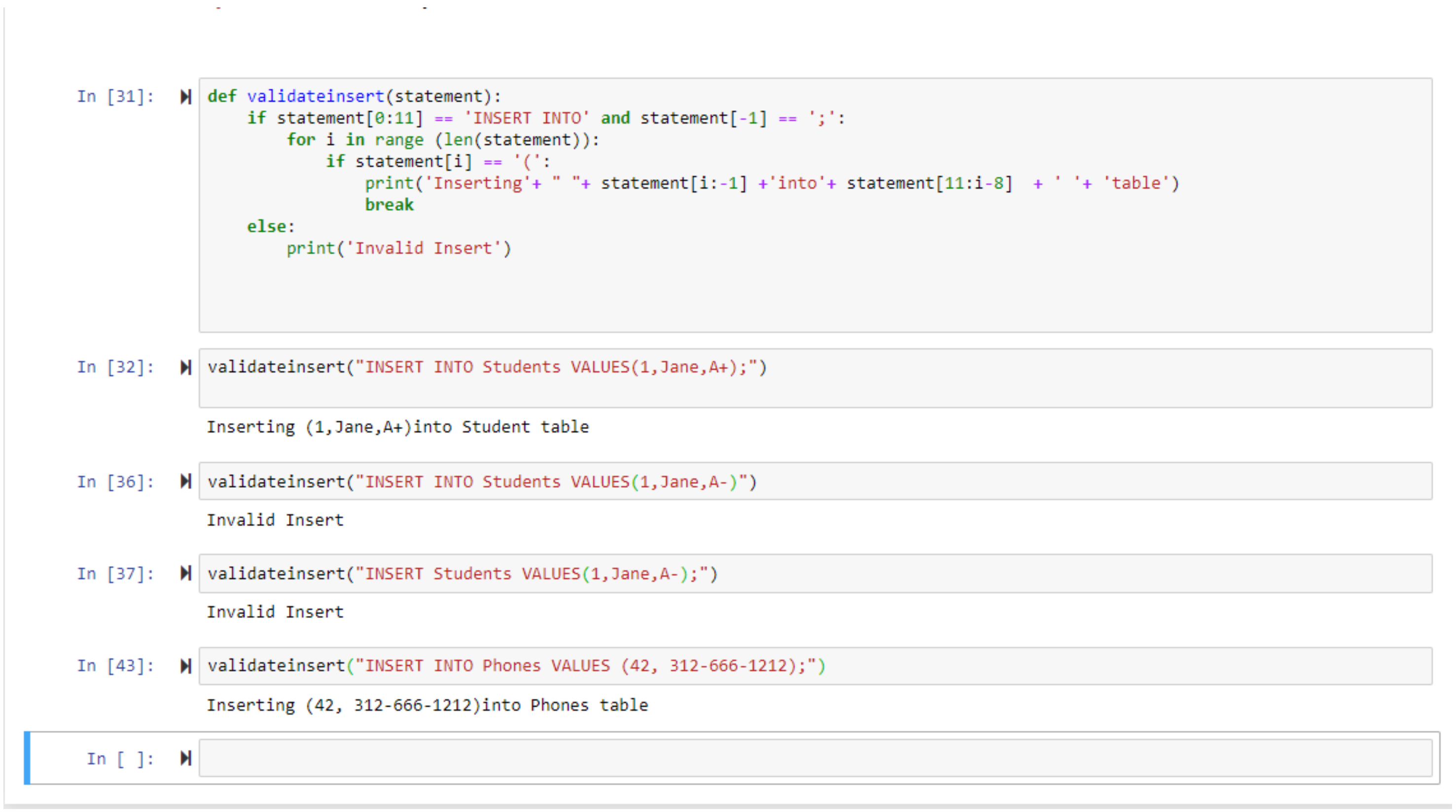
validating the statement, you only have to check that the insert statement starts with INSERT INTO and that statements ends with a semicolon (;).

Examples:
validateInsert(”**INSERT INTO Students VALUES (1, 'Jane', 'A-');**”)
- Inserting (1, 'Jane', 'A+') into Students table

validateInsert(“**INSERT INTO Students VALUES (1, 'Jane', 'A-')**”)
- Invalid insert

validateInsert(“**INSERT Students VALUES (1, 'Jane', A-);**”)
- Invalid insert

validateInsert(“**INSERT INTO Phones VALUES (42, '312-666-1212');**”)
- Inserting (42, '312-666-1212') into Phones table



```
In [31]:  def validateinsert(statement):
              if statement[0:11] == 'INSERT INTO' and statement[-1] == ';':
                  for i in range (len(statement)):
                      if statement[i] == '(':
                          print('Inserting'+ " "+ statement[i:-1] +'into'+ statement[11:i-8]  + ' '+ 'table')
                          break
              else:
                  print('Invalid Insert')

In [32]:  validateinsert("INSERT INTO Students VALUES(1,Jane,A+);")

          Inserting (1,Jane,A+)into Student table

In [36]:  validateinsert("INSERT INTO Students VALUES(1,Jane,A-)")

          Invalid Insert

In [37]:  validateinsert("INSERT Students VALUES(1,Jane,A-);")

          Invalid Insert

In [43]:  validateinsert("INSERT INTO Phones VALUES (42, 312-666-1212);")

          Inserting (42, 312-666-1212)into Phones table

In [ ]:
```

Part 2

Consider a MEETING table that records information about meetings between clients and executives in the company. Each record contains the names of the client and the executive’s name as well as the office number, floor and the building. Finally, each record contains the city that the building is in and the date of the meeting. The table is in First Normal Form and the primary key is (Client, Office).
(Date, <u>Client</u>, <u>Office</u>, Floor, Building, City, Executive)

You are given the following functional dependencies:
Building → City
Office → Floor, Building, City
Client → Executive
Client, Office → Date

a. **For the functional dependency Building → City, explain the redundancy problem and possible consequences through an example (you can make up your own building names as you see fit).**

The Redundancy problem with "Building -City" is that a building name can exist in more than one city. It isn't limited to just one. This could be problematic because, in the case of a building like Trump Tower, a person in New York or Chicago can say they want to meet at the Trump Tower — the problem is that Trump Tower has buildings in both locations. As a result, they will be unable to meet unless otherwise defined, implying that there is no one-of-a-kind city attached to a certain structure.

b. **Remove any existing partial dependencies and convert the logical schema to the Second Normal Form. Please remember that when performing schema decomposition you need to denote primary key for every new table as well as the foreign key that will allow us to reconstruct the original data.**

Remove Partial Dependencies:

- Office –Floor, Building, City
- Client –Executive

Second Normal Form (2NF):

- (Office, Floor, Building, City) ………. **Primary Key:** Office
- (Client, Executive) …………………. **Primary Key:** Client
- (Client, Office, Date) ……………….**Primary Key + Foreign Key:** Client & Office

c. **Remove any existing transitive dependencies to create a set of logical schemas in Third Normal Form. Again, remember to denote primary keys and foreign keys (including which primary key those foreign keys point to).**

Remove Transitive Dependencies:

- Building ⍰City

Third Normal Form (3NF):

- (Building, City)

- (Office, Floor, Building) ……………. **Primary Key:** Office - **Foreign Key:** Building
- (Client, Executive) …………………. **Primary Key:** Client
- (Client, Office, Date) ………………. **Primary Key + Foreign Key:** Client & Office

Part 3

Consider a table that stores information about students, student name, GPA, honors list and the credits that the student had completed so far.

(First, Last, GPA, Honor, Credits)

You are given the following functional dependencies

First, Last → GPA, Honor, Credits
GPA → Honor

a. Is this schema in Second Normal Form? If not, please state which FDs violate 2NF and decompose the schema accordingly.

Yes, the schema above is in Second Normal Form (2NF) given that there are no partial dependencies.

b. Is this schema in Third Normal Form? If not, please state which FDs violate 3NF and decompose the schema accordingly.

No, the above schema is not in Third Normal Form (3NF), as the Honor is now determined by the GPA. This is not valid, because GPA is not a critical attribute like "First" and "Last."

Remove:
- GPA –Honor

Third Normal Form (3NF):
- (GPA, Honor)
- (First, Last, GPA, Credits)