Foundations of Databases - Final Project, Part 4

Case Scenario - Public Library (Circulation)

Umma Islam

25FL-KG573-101 - Foundations of Database

Due Date:12/03/2025

1. **Case Scenario Summary:** A local community library has hired me to upgrade their traditional paper-based card catalog into a modern, organized digital database system. The library supports a large number of members who regularly check out books, and its current manual process makes it difficult to track borrowing, returning, and overdue items efficiently. In this library, a single book title may be written by one author or several authors. At the same time, the library owns multiple physical copies of each title, and every copy must be identifiable through its own unique barcode. When a member checks out a copy, the system needs to create a loan record that stores key details such as the checkout date and the due date. If a member returns a book after its due date, the system must automatically capture and store late fees or fines. Because day-to-day operations involve several library employees, the new database should also maintain information about the librarians who handle checkouts, returns, and manage member interactions. To support all of these activities, the system must organize and relate several different kinds of information, including members, authors, book titles, the relationships between titles and authors, physical book copies, active and past loans, fines that arise from overdue returns, and the librarians who oversee the process. The goal is to replace the old card system with a reliable, searchable, and efficient digital solution that streamlines library operations and improves both accuracy and service. The purpose of the database is to organize and connect all information about members, authors, book titles, copies, loans, fines, and librarians so the library can manage borrowing and inventory more efficiently.

2. Logical Design (Part 1):

The Entity, Descriptions and Attributes		
Entities	**Description**	**Attributes**
1.Member	Represents those who have registered with the library and are eligible to check out books.	Member_ID (Primary Key) First_name Last_name Email Phone Join_Date Status: Active,Suspended or Inactive
2.Librarian	Represents the library employees in charge of overseeing the loan and return procedures.	Librarian_ID(Primary Key) First_name Last_name Email Phone Hire_Date
3.Title	Represents each book title in the library’s catalog.	Title_id (PK) Isbn (UNIQUE), Title publisher publication_year genre
4.Title_Author	Represents books of the name that are provided by the author.	Title_id (FK → TITLE) Author_id (FK → AUTHOR) Primary_author (Y/N)
5.Author	Representing the person who wrote the book.	Author_ID (Primary Key) First_name Last_name
6.loan	Represents who the members Borrow or rent a book from the library.	Loan_id (PK) Copy_id (FK → BOOK_COPY) Member_id (FK → MEMBER) Checkout_by (FK → LIBRARIAN) Checkin_by (FK → LIBRARIAN, optional)’ Checkout_date Due_date

The Entity, Descriptions and Attributes		
		Return_date Status
7.Fine	Represents fines for past-due book returns associated with a particular loan.	Fine_ID(Primary Key) Loan_ID(Foreign Key) Fine_amount Assessed_date Paid_date Status, notes
8. Book copy	Represents every book copy for tracking purposes.	Copy_id (PK) Title_id (FK → TITLE) Barcode (UNIQUE) Copy_number Acquired_date Condition_status Shelf_location

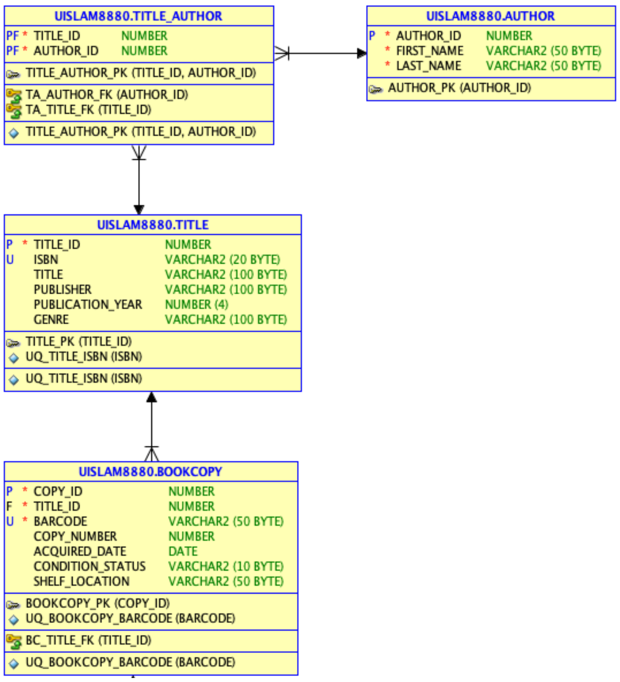
Relationships, Key attributes and Constraints		
Relationships & Cardinality	Key Attributes	Constraints
Title to Bookcopy 1 to Many	Title_id Book_copy	One title can have multiple copies.
Title to Author Many to Recursive	Title_ID Author_ID	Use title_Author as an intersection.
Member to Loan 1 to many	Loan_id (PK) Member_id (PK)	A member can have multiple loans.

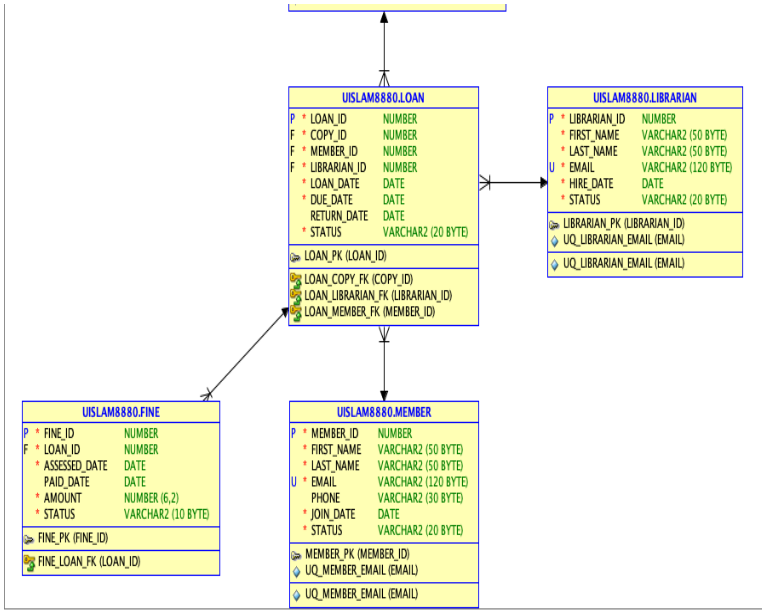
Book copy to Loan
1 to Many
	copy_id(pk)
	loan_id(pk)
		Each copy can be loaned multiple times.

Librarian to Loan
1 to many
	library_id(PK)
	Loan_id(PK)
		Librarians manage the checkout/check-in process

Loan to Fine
1 to 1
	Loan_ID(PK)
	Fine_ID(PK)
		A loan may have zero or one fine

3.Physical Design (Part 2):


UISLAM8880.TITLE_AUTHOR
PF * TITLE_ID NUMBER
PF * AUTHOR_ID NUMBER
TITLE_AUTHOR_PK (TITLE_ID, AUTHOR_ID)
TA_AUTHOR_FK (AUTHOR_ID)
TA_TITLE_FK (TITLE_ID)
TITLE_AUTHOR_PK (TITLE_ID, AUTHOR_ID)
UISLAM8880.AUTHOR
P * AUTHOR_ID NUMBER
* FIRST_NAME VARCHAR2 (50 BYTE)
* LAST_NAME VARCHAR2 (50 BYTE)
AUTHOR_PK (AUTHOR_ID)
UISLAM8880.TITLE
P * TITLE_ID NUMBER
U ISBN VARCHAR2 (20 BYTE)
TITLE VARCHAR2 (100 BYTE)
PUBLISHER VARCHAR2 (100 BYTE)
PUBLICATION_YEAR NUMBER (4)
GENRE VARCHAR2 (100 BYTE)
TITLE_PK (TITLE_ID)
UQ_TITLE_ISBN (ISBN)
UQ_TITLE_ISBN (ISBN)
UISLAM8880.BOOKCOPY
P * COPY_ID NUMBER
F * TITLE_ID NUMBER
U * BARCODE VARCHAR2 (50 BYTE)
COPY_NUMBER NUMBER
ACQUIRED_DATE DATE
CONDITION_STATUS VARCHAR2 (10 BYTE)
SHELF_LOCATION VARCHAR2 (50 BYTE)
BOOKCOPY_PK (COPY_ID)
UQ_BOOKCOPY_BARCODE (BARCODE)
BC_TITLE_FK (TITLE_ID)
UQ_BOOKCOPY_BARCODE (BARCODE)



DDL Scripts:

```
-- Generated by Oracle SQL Developer Data Modeler 24.3.1.351.0831

--   at:        2025-11-29 00:33:57 EST

--   site:      Oracle Database 21c
--   type:      Oracle Database 21c
-- predefined type, no DDL - MDSYS.SDO_GEOMETRY
-- predefined type, no DDL - XMLTYPE
CREATE TABLE UISLAM8880.AUTHOR
   (
    AUTHOR_ID  NUMBER  NOT NULL ,
    FIRST_NAME VARCHAR2 (50 BYTE)  NOT NULL ,
    LAST_NAME  VARCHAR2 (50 BYTE)  NOT NULL
   )
;
ALTER TABLE UISLAM8880.AUTHOR
   ADD CONSTRAINT AUTHOR_PK PRIMARY KEY ( AUTHOR_ID ) ;

CREATE TABLE UISLAM8880.BOOKCOPY
   (
    COPY_ID         NUMBER  NOT NULL ,
    TITLE_ID        NUMBER  NOT NULL ,
    BARCODE         VARCHAR2 (50 BYTE)  NOT NULL ,
    COPY_NUMBER     NUMBER ,
    ACQUIRED_DATE   DATE ,
    CONDITION_STATUS VARCHAR2 (10 BYTE) ,
    SHELF_LOCATION   VARCHAR2 (50 BYTE)
   )
```

```
;
ALTER TABLE UISLAM8880.BOOKCOPY
   ADD CONSTRAINT CK_BOOKCOPY_CONDITION
   CHECK (CONDITION_STATUS IN ('FAIR', 'GOOD', 'NEW', 'POOR'))
;
CREATE UNIQUE INDEX UISLAM8880.UQ_BOOKCOPY_BARCODE ON
UISLAM8880.BOOKCOPY
   (
    BARCODE ASC
   )
;

ALTER TABLE UISLAM8880.BOOKCOPY
CREATE TABLE UISLAM8880.BOOKCOPY
   (
    COPY_ID          NUMBER  NOT NULL ,
    TITLE_ID         NUMBER  NOT NULL ,
    BARCODE          VARCHAR2 (50 BYTE)  NOT NULL ,
    COPY_NUMBER      NUMBER ,
    ACQUIRED_DATE    DATE ,
    CONDITION_STATUS VARCHAR2 (10 BYTE) ,
    SHELF_LOCATION   VARCHAR2 (50 BYTE)
   )
```

```
;
ALTER TABLE UISLAM8880.BOOKCOPY
   ADD CONSTRAINT CK_BOOKCOPY_CONDITION
   CHECK (CONDITION_STATUS IN ('FAIR', 'GOOD', 'NEW', 'POOR'))
;
CREATE UNIQUE INDEX UISLAM8880.UQ_BOOKCOPY_BARCODE ON
UISLAM8880.BOOKCOPY
   (
    BARCODE ASC
   )
;

ALTER TABLE UISLAM8880.BOOKCOPY
   ADD CONSTRAINT BOOKCOPY_PK PRIMARY KEY ( COPY_ID ) ;
ALTER TABLE UISLAM8880.BOOKCOPY
   ADD CONSTRAINT UQ_BOOKCOPY_BARCODE UNIQUE ( BARCODE ) ;
CREATE TABLE UISLAM8880.FINE
   (
    FINE_ID       NUMBER  NOT NULL ,
    LOAN_ID       NUMBER  NOT NULL ,
    ASSESSED_DATE DATE  NOT NULL ,
    PAID_DATE     DATE ,
    AMOUNT        NUMBER (6,2)  NOT NULL ,
    STATUS        VARCHAR2 (10 BYTE) DEFAULT 'UNPAID'  NOT NULL
   )
;
ALTER TABLE UISLAM8880.FINE
   ADD CONSTRAINT CK_FINE_STATUS
   CHECK (STATUS IN ('PAID', 'UNPAID', 'WAIVED'))
;
ALTER TABLE UISLAM8880.FINE
   ADD CONSTRAINT FINE_PK PRIMARY KEY ( FINE_ID ) ;
CREATE TABLE UISLAM8880.LIBRARIAN
   (
    LIBRARIAN_ID NUMBER  NOT NULL ,
    FIRST_NAME   VARCHAR2 (50 BYTE)  NOT NULL ,
    LAST_NAME    VARCHAR2 (50 BYTE)  NOT NULL ,
    EMAIL        VARCHAR2 (120 BYTE)  NOT NULL ,
    HIRE_DATE    DATE DEFAULT SYSDATE  NOT NULL ,
    STATUS       VARCHAR2 (20 BYTE) DEFAULT 'ACTIVE'  NOT NULL
   )
;
ALTER TABLE UISLAM8880.LIBRARIAN
   ADD CONSTRAINT CK_LIBRARIAN_STATUS
   CHECK (STATUS IN ('ACTIVE', 'INACTIVE'))
;
```

```
CREATE UNIQUE INDEX UISLAM8880.UQ_LIBRARIAN_EMAIL ON
UISLAM8880.LIBRARIAN
   (
    EMAIL ASC
   )
;
ALTER TABLE UISLAM8880.LIBRARIAN
   ADD CONSTRAINT LIBRARIAN_PK PRIMARY KEY ( LIBRARIAN_ID ) ;
ALTER TABLE UISLAM8880.LIBRARIAN
   ADD CONSTRAINT UQ_LIBRARIAN_EMAIL UNIQUE ( EMAIL ) ;
CREATE TABLE UISLAM8880.LOAN
   (
    LOAN_ID      NUMBER  NOT NULL ,
    COPY_ID      NUMBER  NOT NULL ,
    MEMBER_ID    NUMBER  NOT NULL ,
    LIBRARIAN_ID NUMBER  NOT NULL ,
    LOAN_DATE    DATE DEFAULT SYSDATE  NOT NULL ,
    DUE_DATE     DATE  NOT NULL ,
    RETURN_DATE  DATE ,
    STATUS       VARCHAR2 (20 BYTE) DEFAULT 'ON_LOAN'  NOT NULL
   )
;
ALTER TABLE UISLAM8880.LOAN
```

```
   ADD CONSTRAINT CK_LOAN_STATUS
   CHECK (STATUS IN ('LOST', 'ON_LOAN', 'RETURNED'))
;
ALTER TABLE UISLAM8880.LOAN
   ADD CONSTRAINT LOAN_PK PRIMARY KEY ( LOAN_ID ) ;

CREATE TABLE UISLAM8880.MEMBER
   (
    MEMBER_ID  NUMBER  NOT NULL ,
    FIRST_NAME VARCHAR2 (50 BYTE)  NOT NULL ,
    LAST_NAME  VARCHAR2 (50 BYTE)  NOT NULL ,
    EMAIL      VARCHAR2 (120 BYTE)  NOT NULL ,
    PHONE      VARCHAR2 (30 BYTE) ,
    JOIN_DATE  DATE DEFAULT SYSDATE  NOT NULL ,
    STATUS     VARCHAR2 (20 BYTE) DEFAULT 'ACTIVE'  NOT NULL
   )
;
ALTER TABLE UISLAM8880.MEMBER
   ADD CONSTRAINT CK_MEMBER_STATUS
   CHECK (STATUS IN ('ACTIVE', 'INACTIVE', 'SUSPENDED'))
;
CREATE UNIQUE INDEX UISLAM8880.UQ_MEMBER_EMAIL ON
UISLAM8880.MEMBER
```

```
   (
    EMAIL ASC
   )
;
ALTER TABLE UISLAM8880.MEMBER
   ADD CONSTRAINT MEMBER_PK PRIMARY KEY ( MEMBER_ID ) ;
ALTER TABLE UISLAM8880.MEMBER
   ADD CONSTRAINT UQ_MEMBER_EMAIL UNIQUE ( EMAIL ) ;
CREATE TABLE UISLAM8880.TITLE
   (
    TITLE_ID         NUMBER  NOT NULL ,
    ISBN             VARCHAR2 (20 BYTE) ,
    TITLE            VARCHAR2 (100 BYTE) ,
    PUBLISHER        VARCHAR2 (100 BYTE) ,
    PUBLICATION_YEAR NUMBER (4) ,
    GENRE            VARCHAR2 (100 BYTE)
   )
;
CREATE UNIQUE INDEX UISLAM8880.UQ_TITLE_ISBN ON UISLAM8880.TITLE
   (
    ISBN ASC
   )
;
```

```
ALTER TABLE UISLAM8880.TITLE
   ADD CONSTRAINT TITLE_PK PRIMARY KEY ( TITLE_ID ) ;
ALTER TABLE UISLAM8880.TITLE
   ADD CONSTRAINT UQ_TITLE_ISBN UNIQUE ( ISBN ) ;
CREATE TABLE UISLAM8880.TITLE_AUTHOR
   (
    TITLE_ID  NUMBER  NOT NULL ,
    AUTHOR_ID NUMBER  NOT NULL
   )
;
CREATE UNIQUE INDEX UISLAM8880.TITLE_AUTHOR_PK ON
UISLAM8880.TITLE_AUTHOR
   (
    TITLE_ID ASC ,
    AUTHOR_ID ASC
   )
;

ALTER TABLE UISLAM8880.TITLE_AUTHOR
   ADD CONSTRAINT TITLE_AUTHOR_PK PRIMARY KEY ( TITLE_ID, AUTHOR_ID ) ;
ALTER TABLE UISLAM8880.BOOKCOPY
   ADD CONSTRAINT BC_TITLE_FK FOREIGN KEY
```

```
    (
     TITLE_ID
    )
    REFERENCES UISLAM8880.TITLE
    (
     TITLE_ID
    )
;
ALTER TABLE UISLAM8880.FINE
    ADD CONSTRAINT FINE_LOAN_FK FOREIGN KEY
    (
     LOAN_ID
    )
    REFERENCES UISLAM8880.LOAN
    (
     LOAN_ID
    )
;
ALTER TABLE UISLAM8880.LOAN
    ADD CONSTRAINT LOAN_COPY_FK FOREIGN KEY
    (
     COPY_ID
    )
```

```
   REFERENCES UISLAM8880.BOOKCOPY
   (
    COPY_ID
   )
;
ALTER TABLE UISLAM8880.LOAN
   ADD CONSTRAINT LOAN_LIBRARIAN_FK FOREIGN KEY
   (
    LIBRARIAN_ID
   )
   REFERENCES UISLAM8880.LIBRARIAN
   (
    LIBRARIAN_ID
   )
;
ALTER TABLE UISLAM8880.LOAN
   ADD CONSTRAINT LOAN_MEMBER_FK FOREIGN KEY
   (
    MEMBER_ID
   )
   REFERENCES UISLAM8880.MEMBER
   (
    MEMBER_ID
   )
;
ALTER TABLE UISLAM8880.TITLE_AUTHOR
   ADD CONSTRAINT TA_AUTHOR_FK FOREIGN KEY
   (
    AUTHOR_ID
   )
   REFERENCES UISLAM8880.AUTHOR
   (
    AUTHOR_ID
   )
;

ALTER TABLE UISLAM8880.TITLE_AUTHOR
   ADD CONSTRAINT TA_TITLE_FK FOREIGN KEY
   (
    TITLE_ID
   )
   REFERENCES UISLAM8880.TITLE
   (
    TITLE_ID
   )
;
```

```
CREATE SEQUENCE UISLAM8880.AUTHOR_AUTHOR_ID_SEQ
START WITH 1
   CACHE 20 ;
CREATE OR REPLACE TRIGGER UISLAM8880.AUTHOR_AUTHOR_ID_TRG
BEFORE INSERT ON UISLAM8880.AUTHOR
FOR EACH ROW
BEGIN
   :NEW.AUTHOR_ID := UISLAM8880.AUTHOR_AUTHOR_ID_SEQ.NEXTVAL;
END;
/
CREATE SEQUENCE UISLAM8880.BOOKCOPY_COPY_ID_SEQ
START WITH 1
   CACHE 20 ;
CREATE OR REPLACE TRIGGER UISLAM8880.BOOKCOPY_COPY_ID_TRG
BEFORE INSERT ON UISLAM8880.BOOKCOPY
FOR EACH ROW
BEGIN
   :NEW.COPY_ID := UISLAM8880.BOOKCOPY_COPY_ID_SEQ.NEXTVAL;
END;
/
CREATE SEQUENCE UISLAM8880.FINE_FINE_ID_SEQ
START WITH 1
   CACHE 20 ;
```

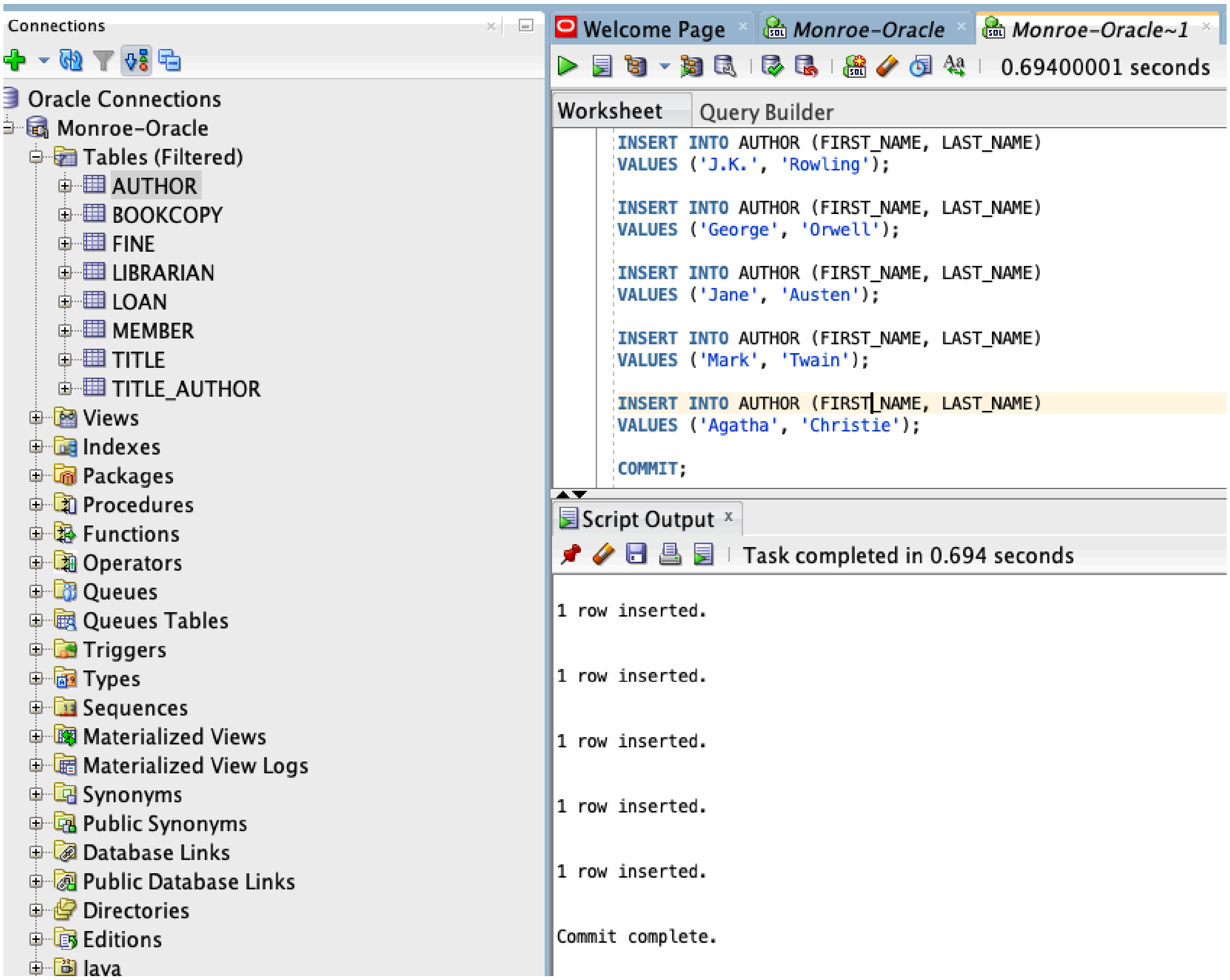
```
CREATE OR REPLACE TRIGGER UISLAM8880.FINE_FINE_ID_TRG
BEFORE INSERT ON UISLAM8880.FINE
FOR EACH ROW
BEGIN
   :NEW.FINE_ID := UISLAM8880.FINE_FINE_ID_SEQ.NEXTVAL;
END;
/
CREATE SEQUENCE UISLAM8880.LIBRARIAN_LIBRARIAN_ID_SEQ
START WITH 1
   CACHE 20 ;
CREATE OR REPLACE TRIGGER UISLAM8880.LIBRARIAN_LIBRARIAN_ID_TRG
BEFORE INSERT ON UISLAM8880.LIBRARIAN
FOR EACH ROW
BEGIN
   :NEW.LIBRARIAN_ID := UISLAM8880.LIBRARIAN_LIBRARIAN_ID_SEQ.NEXTVAL;
END;
/
CREATE SEQUENCE UISLAM8880.LOAN_LOAN_ID_SEQ
START WITH 1
   CACHE 20 ;
CREATE OR REPLACE TRIGGER UISLAM8880.LOAN_LOAN_ID_TRG
BEFORE INSERT ON UISLAM8880.LOAN
FOR EACH ROW
```

```
BEGIN
   :NEW.LOAN_ID := UISLAM8880.LOAN_LOAN_ID_SEQ.NEXTVAL;
END;
/
CREATE SEQUENCE UISLAM8880.MEMBER_MEMBER_ID_SEQ
START WITH 1
   CACHE 20 ;
CREATE OR REPLACE TRIGGER UISLAM8880.MEMBER_MEMBER_ID_TRG
BEFORE INSERT ON UISLAM8880.MEMBER
FOR EACH ROW
BEGIN
   :NEW.MEMBER_ID := UISLAM8880.MEMBER_MEMBER_ID_SEQ.NEXTVAL;
END;
/
CREATE SEQUENCE UISLAM8880.TITLE_TITLE_ID_SEQ
START WITH 1
   CACHE 20 ;
CREATE OR REPLACE TRIGGER UISLAM8880.TITLE_TITLE_ID_TRG
BEFORE INSERT ON UISLAM8880.TITLE
FOR EACH ROW
BEGIN
   :NEW.TITLE_ID := UISLAM8880.TITLE_TITLE_ID_SEQ.NEXTVAL;
END;
```

```
/

-- Oracle SQL Developer Data Modeler Summary Report:

--

-- CREATE TABLE                            8

-- CREATE INDEX                            5

-- ALTER TABLE                            24

-- CREATE VIEW                             0

-- ALTER VIEW                              0

-- CREATE PACKAGE                          0

-- CREATE PACKAGE BODY                     0

-- CREATE PROCEDURE                        0

-- CREATE FUNCTION                         0

-- CREATE TRIGGER                          7

-- ALTER TRIGGER                           0

-- CREATE COLLECTION TYPE                  0

-- CREATE STRUCTURED TYPE                  0

-- CREATE STRUCTURED TYPE BODY             0

-- CREATE CLUSTER                          0

-- CREATE CONTEXT                          0

-- CREATE DATABASE                         0

-- CREATE DIMENSION                        0

-- CREATE DIRECTORY                        0

-- CREATE DISK GROUP                       0
-- CREATE ROLE                               0
-- CREATE ROLLBACK SEGMENT                   0
-- CREATE SEQUENCE                           7
-- CREATE MATERIALIZED VIEW                  0
-- CREATE MATERIALIZED VIEW LOG              0
-- CREATE SYNONYM                            0
-- CREATE TABLESPACE                         0
-- CREATE USER                               0
--
-- DROP TABLESPACE                           0
-- DROP DATABASE                             0
--
-- REDACTION POLICY                          0
--
-- ORDS DROP SCHEMA                          0
-- ORDS ENABLE SCHEMA                        0
-- ORDS ENABLE OBJECT                        0
--
-- ERRORS                                    0
-- WARNINGS                                  0
```

Sample Data:

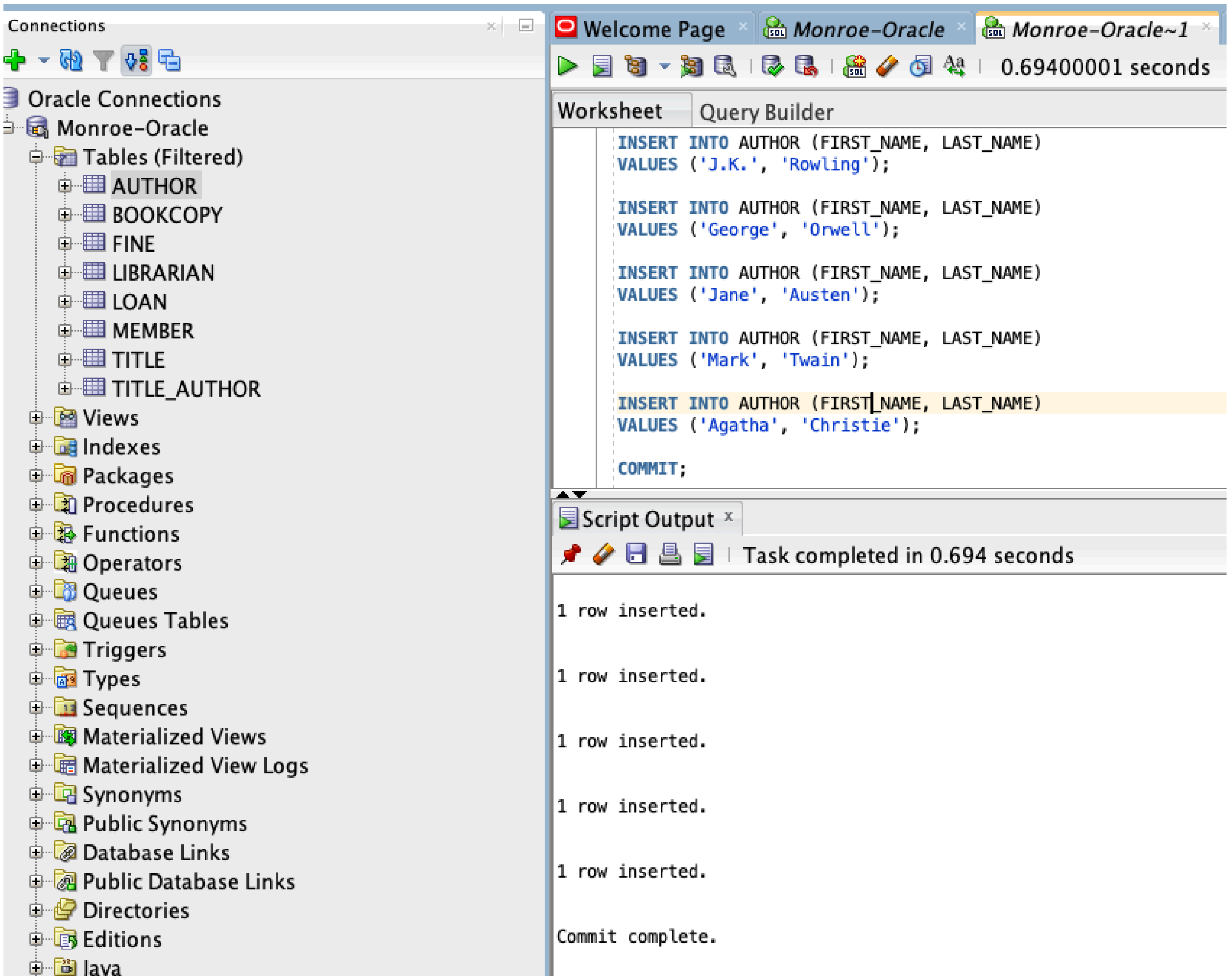
1.



```sql
INSERT INTO AUTHOR (FIRST_NAME, LAST_NAME)
VALUES ('J.K.', 'Rowling');

INSERT INTO AUTHOR (FIRST_NAME, LAST_NAME)
VALUES ('George', 'Orwell');

INSERT INTO AUTHOR (FIRST_NAME, LAST_NAME)
VALUES ('Jane', 'Austen');

INSERT INTO AUTHOR (FIRST_NAME, LAST_NAME)
VALUES ('Mark', 'Twain');

INSERT INTO AUTHOR (FIRST_NAME, LAST_NAME)
VALUES ('Agatha', 'Christie');

COMMIT;
```

```
1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


Commit complete.
```

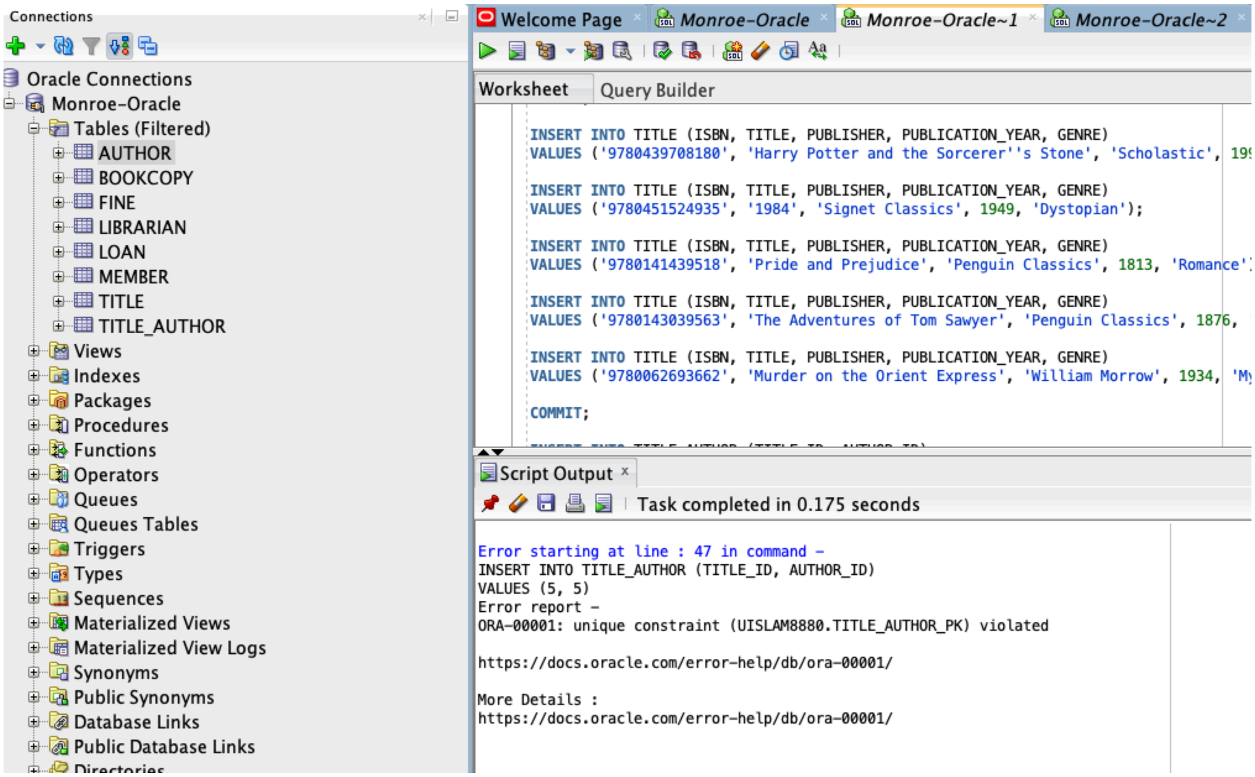
2.



```
INSERT INTO AUTHOR (FIRST_NAME, LAST_NAME)
VALUES ('J.K.', 'Rowling');

INSERT INTO AUTHOR (FIRST_NAME, LAST_NAME)
VALUES ('George', 'Orwell');

INSERT INTO AUTHOR (FIRST_NAME, LAST_NAME)
VALUES ('Jane', 'Austen');

INSERT INTO AUTHOR (FIRST_NAME, LAST_NAME)
VALUES ('Mark', 'Twain');

INSERT INTO AUTHOR (FIRST_NAME, LAST_NAME)
VALUES ('Agatha', 'Christie');

COMMIT;
```

```
1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


Commit complete.
```

3.

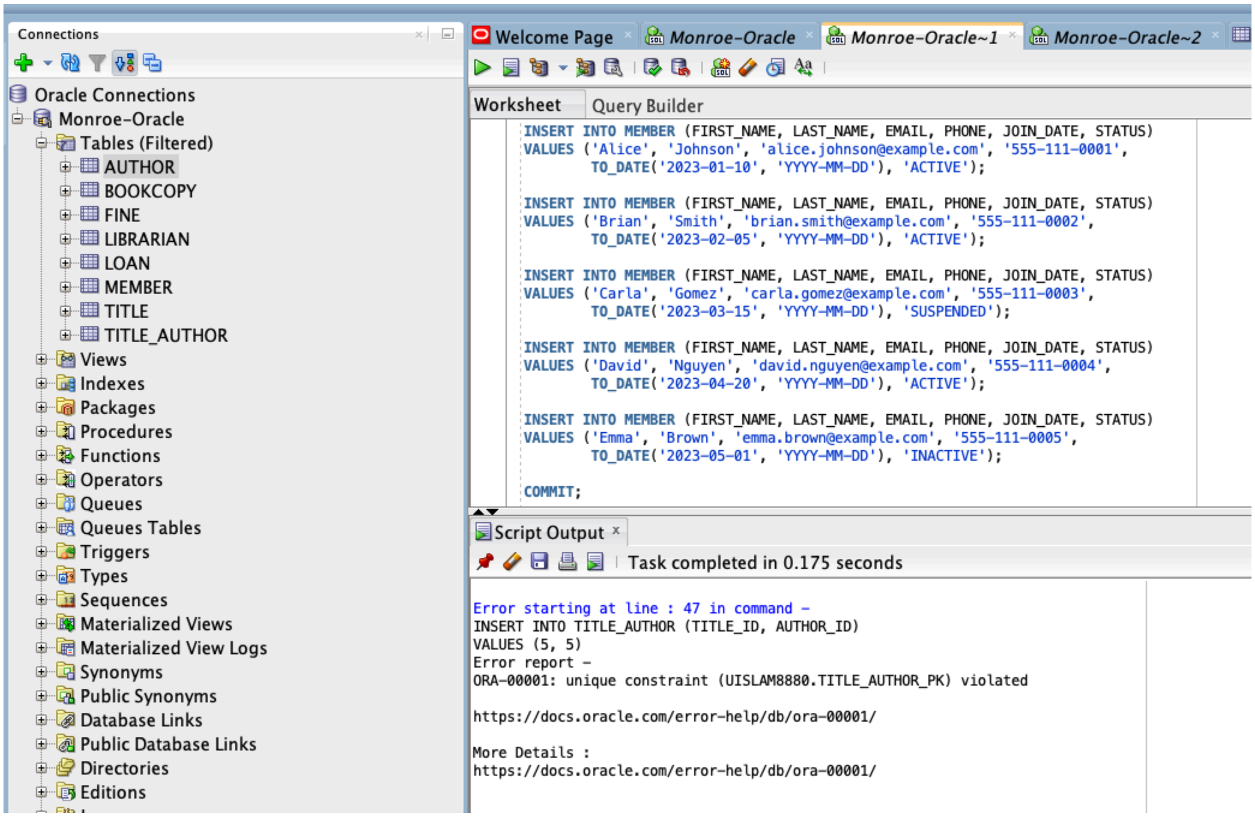


```
INSERT INTO TITLE (ISBN, TITLE, PUBLISHER, PUBLICATION_YEAR, GENRE)
VALUES ('9780439708180', 'Harry Potter and the Sorcerer''s Stone', 'Scholastic', 19

INSERT INTO TITLE (ISBN, TITLE, PUBLISHER, PUBLICATION_YEAR, GENRE)
VALUES ('9780451524935', '1984', 'Signet Classics', 1949, 'Dystopian');

INSERT INTO TITLE (ISBN, TITLE, PUBLISHER, PUBLICATION_YEAR, GENRE)
VALUES ('9780141439518', 'Pride and Prejudice', 'Penguin Classics', 1813, 'Romance'

INSERT INTO TITLE (ISBN, TITLE, PUBLISHER, PUBLICATION_YEAR, GENRE)
VALUES ('9780143039563', 'The Adventures of Tom Sawyer', 'Penguin Classics', 1876,

INSERT INTO TITLE (ISBN, TITLE, PUBLISHER, PUBLICATION_YEAR, GENRE)
VALUES ('9780062693662', 'Murder on the Orient Express', 'William Morrow', 1934, 'My

COMMIT;

INSERT INTO TITLE_AUTHOR (TITLE_ID, AUTHOR_ID)
```

Task completed in 0.175 seconds

```
Error starting at line : 47 in command -
INSERT INTO TITLE_AUTHOR (TITLE_ID, AUTHOR_ID)
VALUES (5, 5)
Error report -
ORA-00001: unique constraint (UISLAM8880.TITLE_AUTHOR_PK) violated

https://docs.oracle.com/error-help/db/ora-00001/

More Details :
https://docs.oracle.com/error-help/db/ora-00001/
```

4.

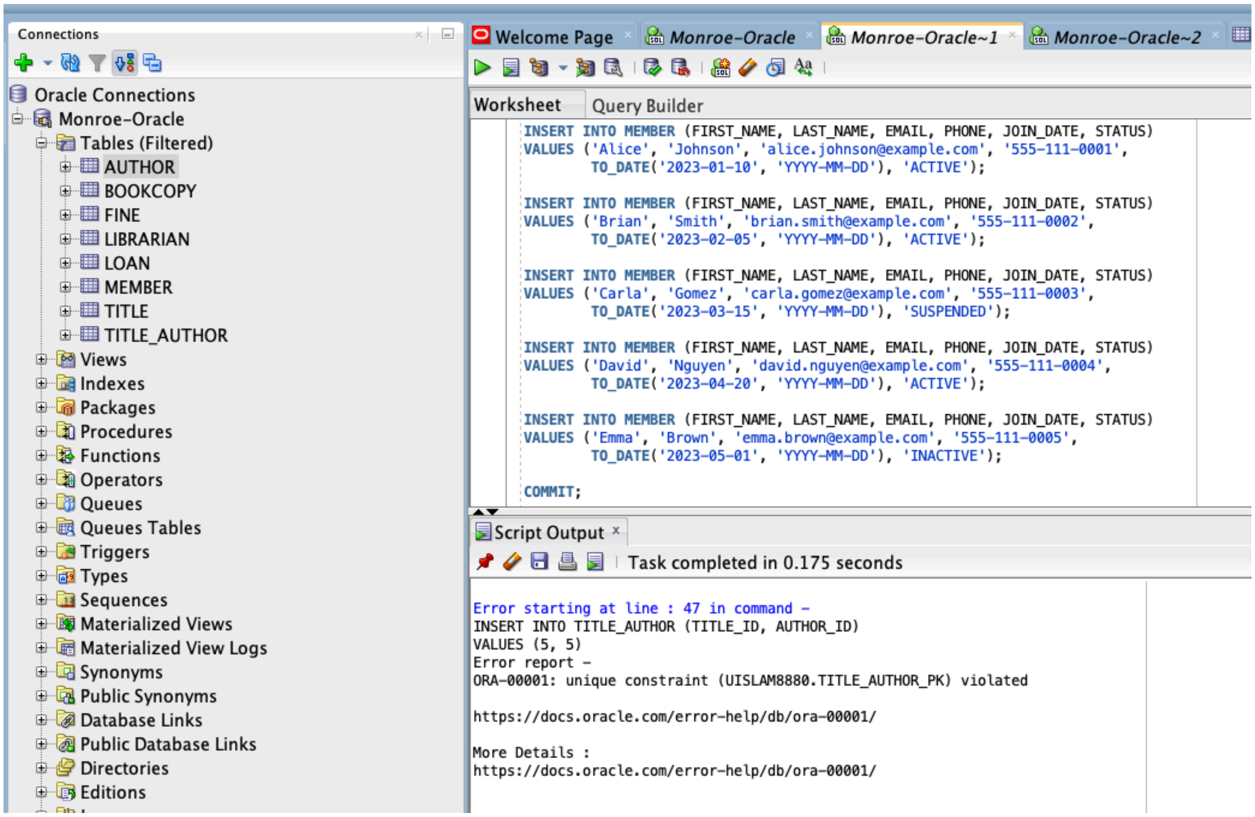


```
INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Alice', 'Johnson', 'alice.johnson@example.com', '555-111-0001',
        TO_DATE('2023-01-10', 'YYYY-MM-DD'), 'ACTIVE');

INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Brian', 'Smith', 'brian.smith@example.com', '555-111-0002',
        TO_DATE('2023-02-05', 'YYYY-MM-DD'), 'ACTIVE');

INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Carla', 'Gomez', 'carla.gomez@example.com', '555-111-0003',
        TO_DATE('2023-03-15', 'YYYY-MM-DD'), 'SUSPENDED');

INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('David', 'Nguyen', 'david.nguyen@example.com', '555-111-0004',
        TO_DATE('2023-04-20', 'YYYY-MM-DD'), 'ACTIVE');

INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Emma', 'Brown', 'emma.brown@example.com', '555-111-0005',
        TO_DATE('2023-05-01', 'YYYY-MM-DD'), 'INACTIVE');

COMMIT;
```

Task completed in 0.175 seconds

```
Error starting at line : 47 in command -
INSERT INTO TITLE_AUTHOR (TITLE_ID, AUTHOR_ID)
VALUES (5, 5)
Error report -
ORA-00001: unique constraint (UISLAM8880.TITLE_AUTHOR_PK) violated

https://docs.oracle.com/error-help/db/ora-00001/

More Details :
https://docs.oracle.com/error-help/db/ora-00001/
```

5.



```
INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Alice', 'Johnson', 'alice.johnson@example.com', '555-111-0001',
        TO_DATE('2023-01-10', 'YYYY-MM-DD'), 'ACTIVE');

INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Brian', 'Smith', 'brian.smith@example.com', '555-111-0002',
        TO_DATE('2023-02-05', 'YYYY-MM-DD'), 'ACTIVE');

INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Carla', 'Gomez', 'carla.gomez@example.com', '555-111-0003',
        TO_DATE('2023-03-15', 'YYYY-MM-DD'), 'SUSPENDED');

INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('David', 'Nguyen', 'david.nguyen@example.com', '555-111-0004',
        TO_DATE('2023-04-20', 'YYYY-MM-DD'), 'ACTIVE');

INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Emma', 'Brown', 'emma.brown@example.com', '555-111-0005',
        TO_DATE('2023-05-01', 'YYYY-MM-DD'), 'INACTIVE');

COMMIT;
```

```
Error starting at line : 47 in command -
INSERT INTO TITLE_AUTHOR (TITLE_ID, AUTHOR_ID)
VALUES (5, 5)
Error report -
ORA-00001: unique constraint (UISLAM8880.TITLE_AUTHOR_PK) violated

https://docs.oracle.com/error-help/db/ora-00001/

More Details :
https://docs.oracle.com/error-help/db/ora-00001/
```

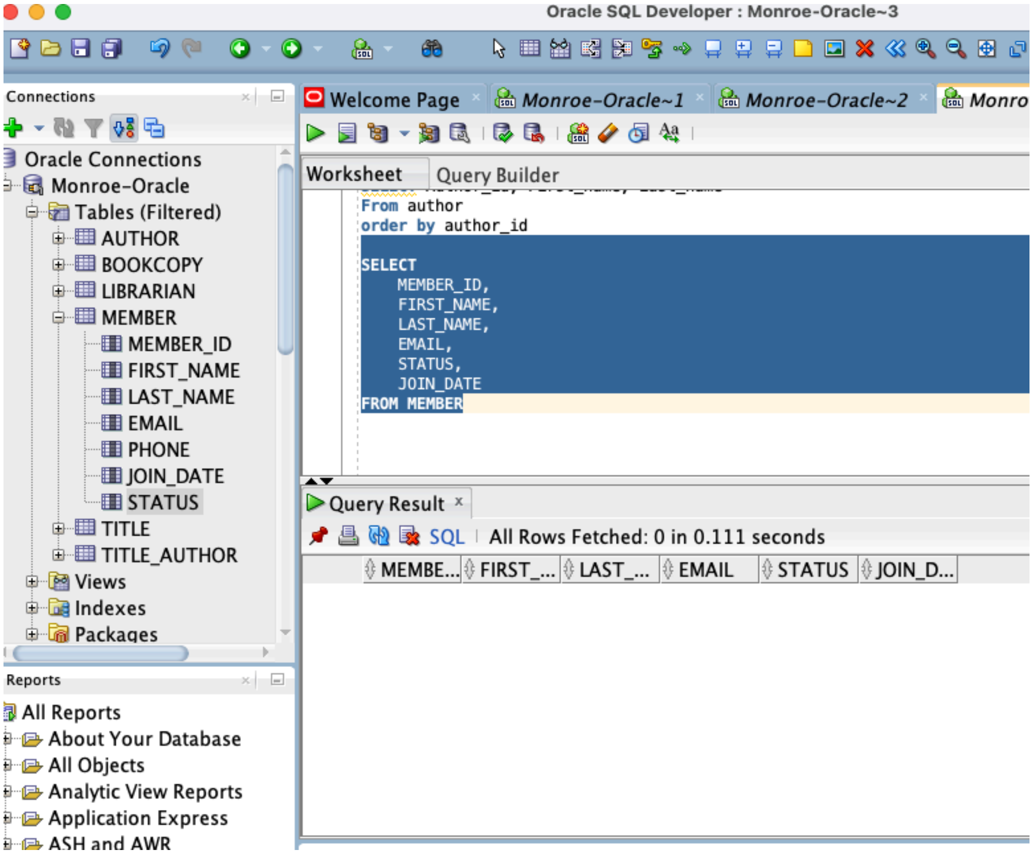
4.SQL Queries (Part 3):

1.Basic Queries (SELECT)

Write at least 3 SELECT statements that use:

▪Specific columns (projection)

```
SELECT
   MEMBER_ID,
   FIRST_NAME,
   LAST_NAME,
   EMAIL,
   STATUS,
   JOIN_DATE
FROM MEMBER
```

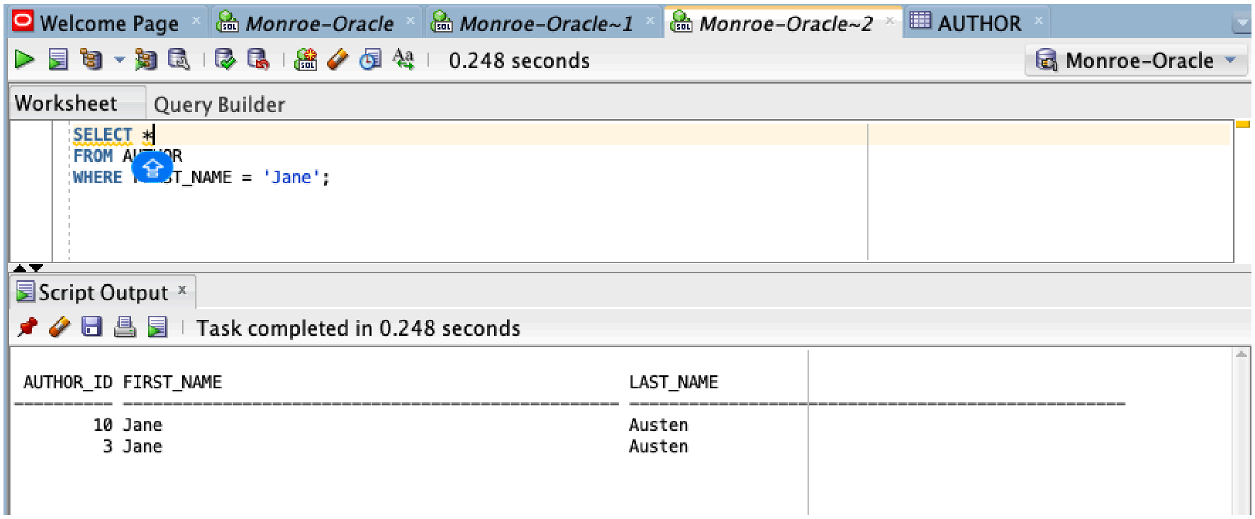


```
From author
order by author_id

SELECT
    MEMBER_ID,
    FIRST_NAME,
    LAST_NAME,
    EMAIL,
    STATUS,
    JOIN_DATE
FROM MEMBER
```

All Rows Fetched: 0 in 0.111 seconds

MEMBE...	FIRST_...	LAST_...	EMAIL	STATUS	JOIN_D...

▪Conditions using WHERE
SELECT *
FROM AUTHOR
WHERE FIRST_NAME = 'Jane';



```
SELECT *
FROM AUTHOR
WHERE FIRST_NAME = 'Jane';
```

Task completed in 0.248 seconds

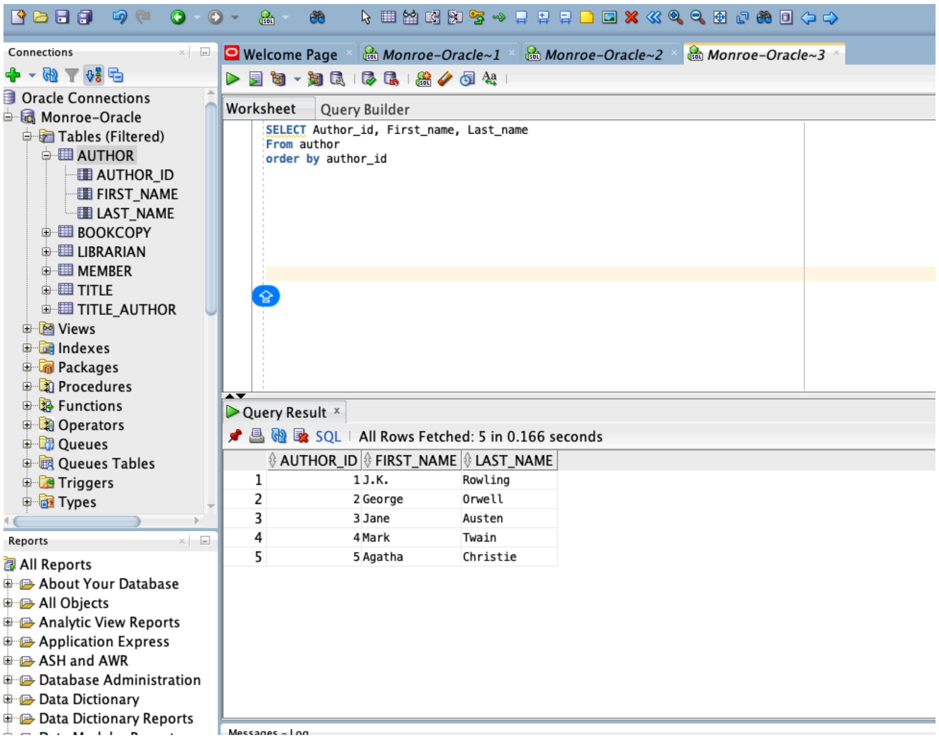
```
AUTHOR_ID FIRST_NAME                                          LAST_NAME
---------- -------------------------------------------------- --------------------------------------------------
        10 Jane                                               Austen
         3 Jane                                               Austen
```

▪Sorting using ORDER BY

SELECT Author_id, First_name, Last_name

From author

order by author_id



```
SELECT Author_id, First_name, Last_name
From author
order by author_id
```

All Rows Fetched: 5 in 0.166 seconds

	AUTHOR_ID	FIRST_NAME	LAST_NAME
1	1	J.K.	Rowling
2	2	George	Orwell
3	3	Jane	Austen
4	4	Mark	Twain
5	5	Agatha	Christie

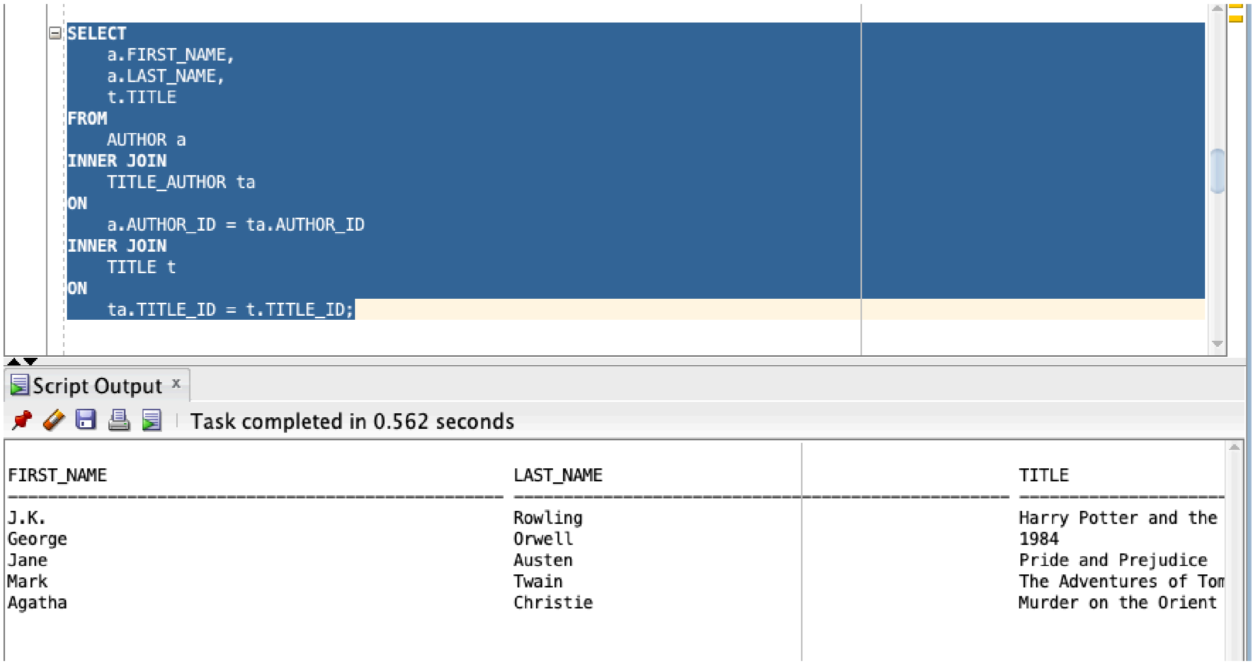
2. Join Queries

Write at least 3 queries that combine data from two or more tables using JOINs.

Examples: INNER JOIN, LEFT JOIN

1. Using Inner join command:

```
SELECT
   a.FIRST_NAME,
   a.LAST_NAME,
   t.TITLE
FROM
   AUTHOR a
INNER JOIN
   TITLE_AUTHOR ta
ON
   a.AUTHOR_ID = ta.AUTHOR_ID
INNER JOIN
   TITLE t
ON
   ta.TITLE_ID = t.TITLE_ID;
```



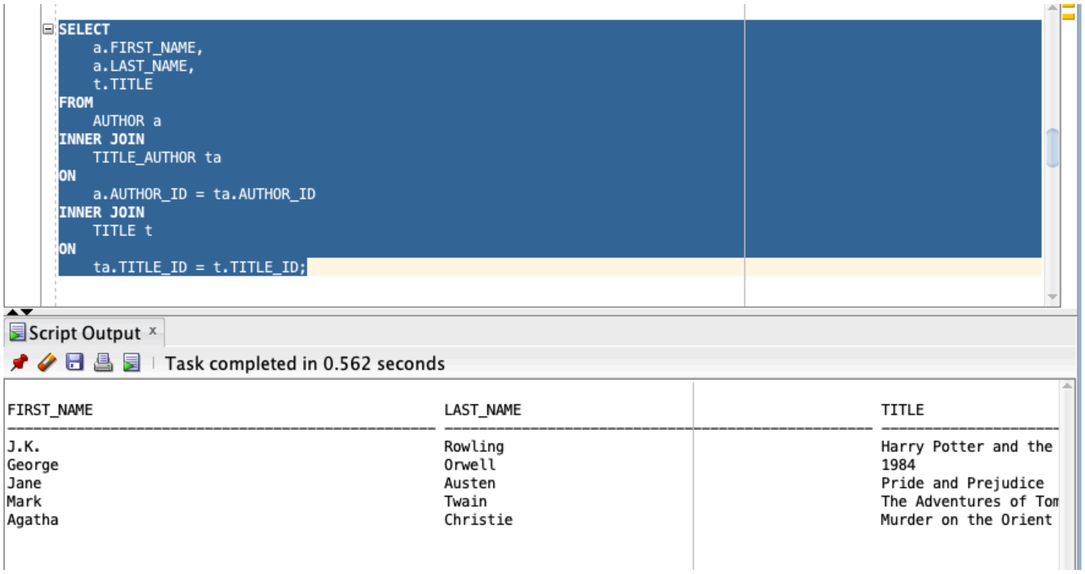
```
SELECT
    a.FIRST_NAME,
    a.LAST_NAME,
    t.TITLE
FROM
    AUTHOR a
INNER JOIN
    TITLE_AUTHOR ta
ON
    a.AUTHOR_ID = ta.AUTHOR_ID
INNER JOIN
    TITLE t
ON
    ta.TITLE_ID = t.TITLE_ID;
```

Task completed in 0.562 seconds

```
FIRST_NAME                                                  LAST_NAME                                                   TITLE
----------------------------------------------------------- ----------------------------------------------------------- -------------------------
J.K.                                                        Rowling                                                     Harry Potter and the
George                                                      Orwell                                                      1984
Jane                                                        Austen                                                      Pride and Prejudice
Mark                                                        Twain                                                       The Adventures of Tom
Agatha                                                      Christie                                                    Murder on the Orient
```

2. Using left join Command:

```
SELECT
   t.TITLE,
   a.FIRST_NAME,
   a.LAST_NAME
FROM  TITLE t
LEFT JOIN TITLE_AUTHOR ta
ON t.TITLE_ID = ta.TITLE_ID
LEFT JOIN  AUTHOR a
ON ta.AUTHOR_ID = a.AUTHOR_ID;
```

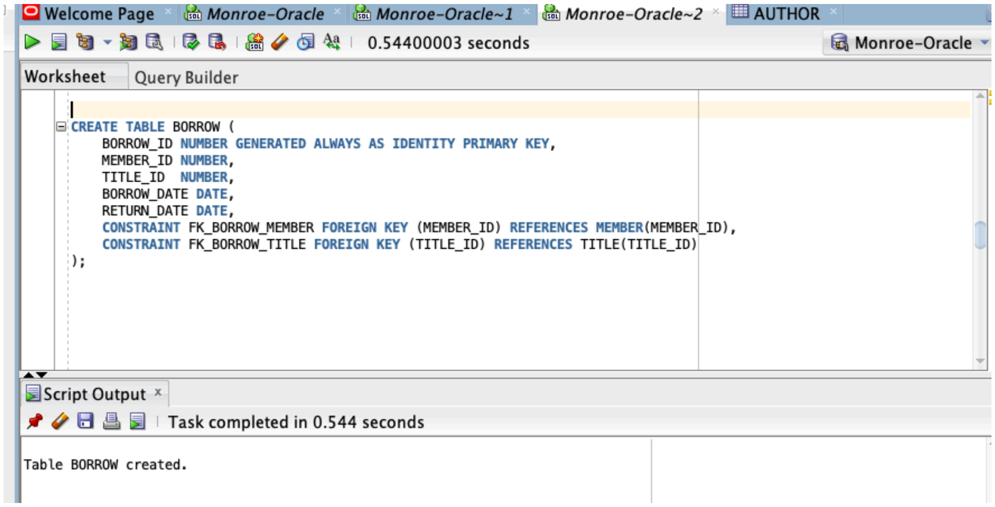


```
SELECT
    a.FIRST_NAME,
    a.LAST_NAME,
    t.TITLE
FROM
    AUTHOR a
INNER JOIN
    TITLE_AUTHOR ta
ON
    a.AUTHOR_ID = ta.AUTHOR_ID
INNER JOIN
    TITLE t
ON
    ta.TITLE_ID = t.TITLE_ID;
```

```
FIRST_NAME                                          LAST_NAME                                                                                                  TITLE
--------------------------------------------------- ---------------------------------------------------------------------------------------------------------- --------------------
J.K.                                                Rowling                                                                                                    Harry Potter and the
George                                              Orwell                                                                                                     1984
Jane                                                Austen                                                                                                     Pride and Prejudice
Mark                                                Twain                                                                                                      The Adventures of Tom
Agatha                                              Christie                                                                                                   Murder on the Orient
```

3.Using Borrow command.

CREATE TABLE BORROW (
    BORROW_ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    MEMBER_ID NUMBER,
    TITLE_ID NUMBER,
    BORROW_DATE DATE,
    RETURN_DATE DATE,
    CONSTRAINT FK_BORROW_MEMBER FOREIGN KEY (MEMBER_ID) REFERENCES MEMBER(MEMBER_ID),
    CONSTRAINT FK_BORROW_TITLE FOREIGN KEY (TITLE_ID) REFERENCES TITLE(TITLE_ID)
);



```
CREATE TABLE BORROW (
    BORROW_ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    MEMBER_ID NUMBER,
    TITLE_ID  NUMBER,
    BORROW_DATE DATE,
    RETURN_DATE DATE,
    CONSTRAINT FK_BORROW_MEMBER FOREIGN KEY (MEMBER_ID) REFERENCES MEMBER(MEMBER_ID),
    CONSTRAINT FK_BORROW_TITLE FOREIGN KEY (TITLE_ID) REFERENCES TITLE(TITLE_ID)
);
```

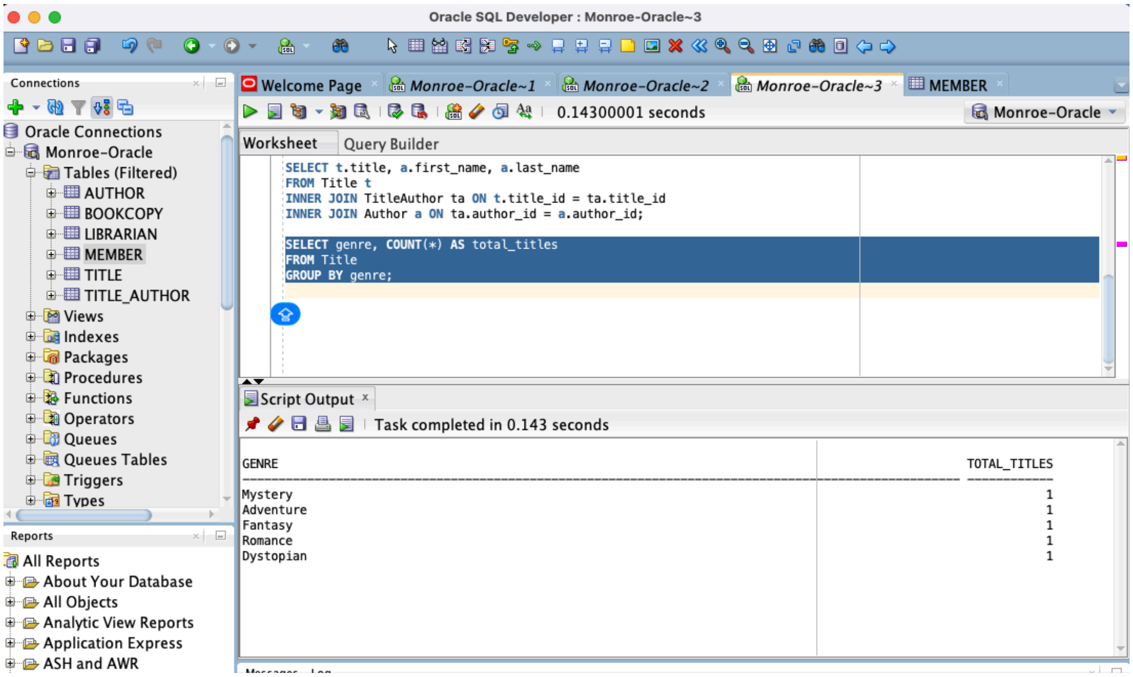
```
Table BORROW created.
```

3.Aggregate Queries

oWrite at least 3 queries using aggregate functions such as COUNT, SUM, AVG, MIN, or MAX.
oUse GROUP BY and HAVING where appropriate.

Group by:

SELECT genre, COUNT(*) AS total_titles
FROM Title
GROUP BY genre;



0.14300001 seconds

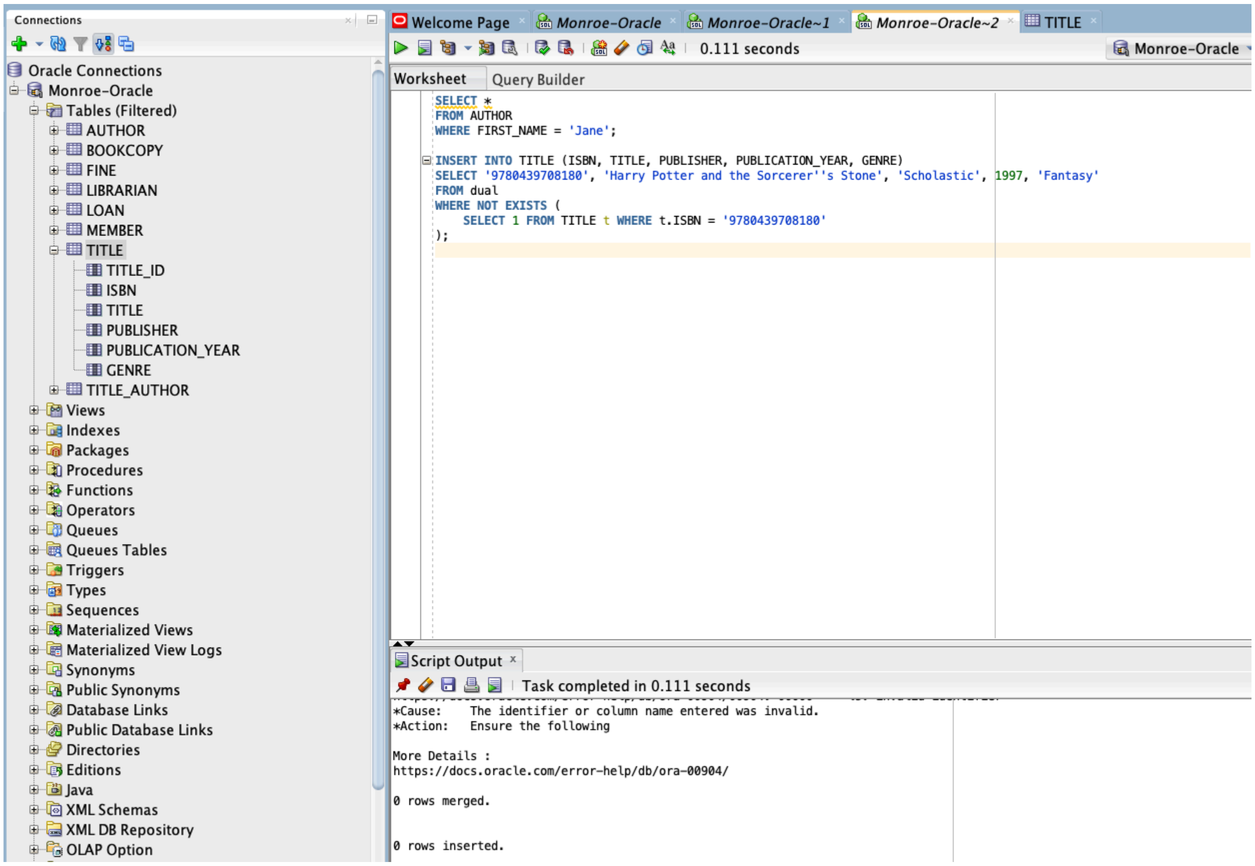
```
SELECT t.title, a.first_name, a.last_name
FROM Title t
INNER JOIN TitleAuthor ta ON t.title_id = ta.title_id
INNER JOIN Author a ON ta.author_id = a.author_id;

SELECT genre, COUNT(*) AS total_titles
FROM Title
GROUP BY genre;
```

Task completed in 0.143 seconds

```
GENRE                                                                                                 TOTAL_TITLES
----------------------------------------------------------------------------------------------------  ------------
Mystery                                                                                                          1
Adventure                                                                                                        1
Fantasy                                                                                                          1
Romance                                                                                                          1
Dystopian                                                                                                        1
```

Count:

SELECT
   t.TITLE,
   COUNT(ta.AUTHOR_ID) AS AUTHOR_COUNT
FROM
   TITLE t
JOIN
   TITLE_AUTHOR ta
ON
   t.TITLE_ID = ta.TITLE_ID
GROUP BY
   t.TITLE
HAVING
   COUNT(ta.AUTHOR_ID) > 1;

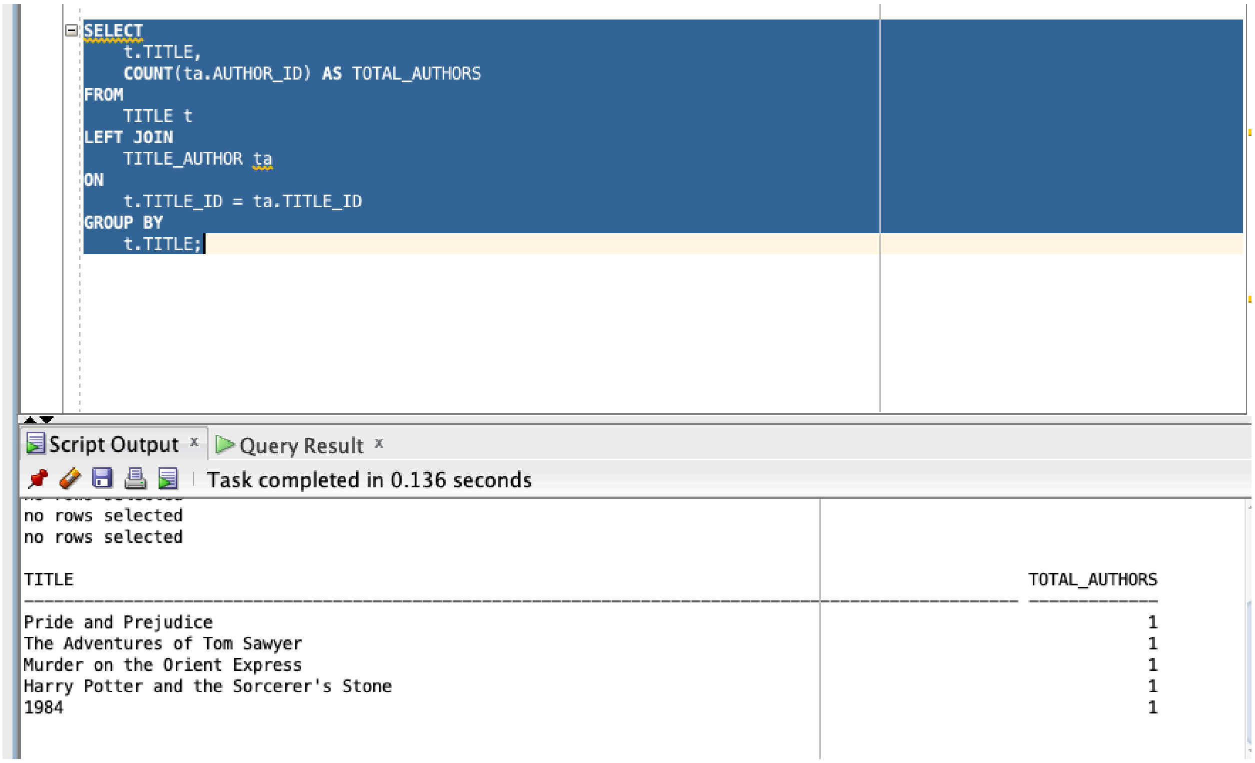


```
SELECT *
FROM AUTHOR
WHERE FIRST_NAME = 'Jane';

INSERT INTO TITLE (ISBN, TITLE, PUBLISHER, PUBLICATION_YEAR, GENRE)
SELECT '9780439708180', 'Harry Potter and the Sorcerer''s Stone', 'Scholastic', 1997, 'Fantasy'
FROM dual
WHERE NOT EXISTS (
    SELECT 1 FROM TITLE t WHERE t.ISBN = '9780439708180'
);
```

```
*Cause:    The identifier or column name entered was invalid.
*Action:   Ensure the following

More Details :
https://docs.oracle.com/error-help/db/ora-00904/

0 rows merged.


0 rows inserted.
```

Sum:

```
SELECT
   t.TITLE,
   COUNT(ta.AUTHOR_ID) AS TOTAL_AUTHORS
FROM
  TITLE t
LEFT JOIN
   TITLE_AUTHOR ta
ON
   t.TITLE_ID = ta.TITLE_ID
GROUP BY
   t.TITLE;
```



```
SELECT
    t.TITLE,
    COUNT(ta.AUTHOR_ID) AS TOTAL_AUTHORS
FROM
    TITLE t
LEFT JOIN
    TITLE_AUTHOR ta
ON
    t.TITLE_ID = ta.TITLE_ID
GROUP BY
    t.TITLE;
```

Task completed in 0.136 seconds

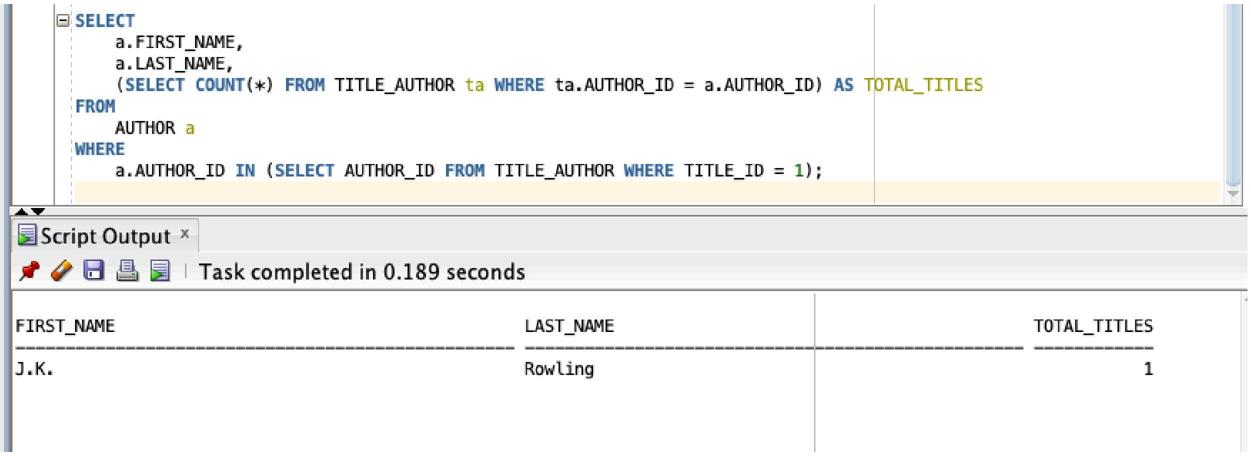
```
no rows selected
no rows selected

TITLE                                                                                                    TOTAL_AUTHORS
-------------------------------------------------------------------------------------------------------- -------------
Pride and Prejudice                                                                                                  1
The Adventures of Tom Sawyer                                                                                         1
Murder on the Orient Express                                                                                         1
Harry Potter and the Sorcerer's Stone                                                                                1
1984                                                                                                                 1
```

4.Subqueries

oWrite at least 2 queries that use subqueries (nested SELECT statements).

1.Using nested SELECT statements

SELECT
   a.FIRST_NAME,
   a.LAST_NAME,
   (SELECT COUNT(*) FROM TITLE_AUTHOR ta WHERE ta.AUTHOR_ID = a.AUTHOR_ID) AS TOTAL_TITLES
FROM
   AUTHOR a
WHERE
   a.AUTHOR_ID IN (SELECT AUTHOR_ID FROM TITLE_AUTHOR WHERE TITLE_ID = 1);

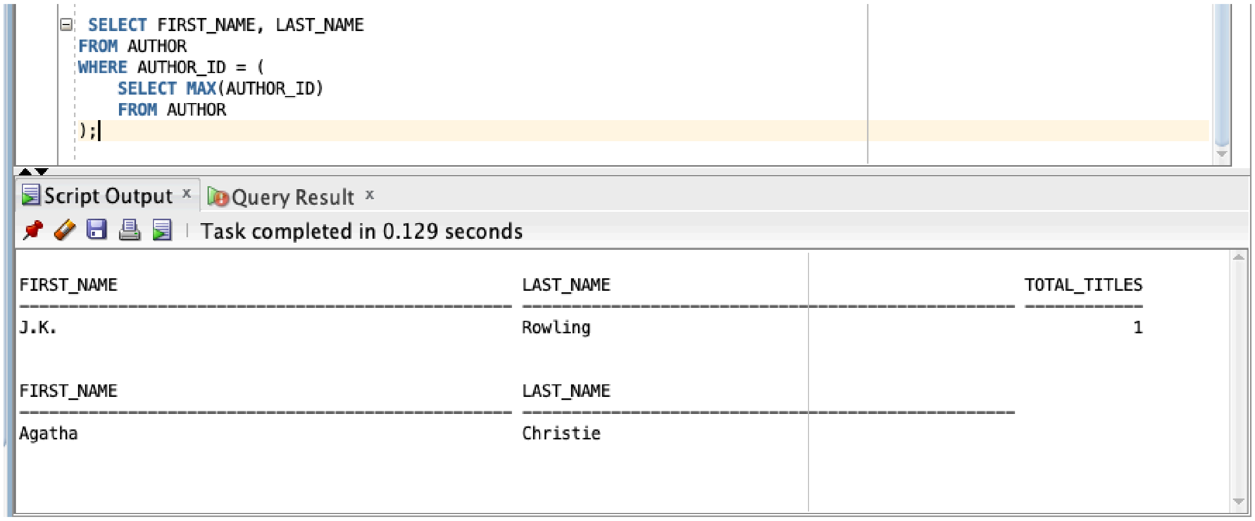


```
SELECT
    a.FIRST_NAME,
    a.LAST_NAME,
    (SELECT COUNT(*) FROM TITLE_AUTHOR ta WHERE ta.AUTHOR_ID = a.AUTHOR_ID) AS TOTAL_TITLES
FROM
    AUTHOR a
WHERE
    a.AUTHOR_ID IN (SELECT AUTHOR_ID FROM TITLE_AUTHOR WHERE TITLE_ID = 1);
```

```
Task completed in 0.189 seconds

FIRST_NAME                                          LAST_NAME                                           TOTAL_TITLES
--------------------------------------------------  --------------------------------------------------  ------------
J.K.                                                Rowling                                                        1
```

2.Using nested SELECT statements

SELECT FIRST_NAME, LAST_NAME
FROM AUTHOR
WHERE AUTHOR_ID = (
    SELECT MAX(AUTHOR_ID)
    FROM AUTHOR
);



```
 SELECT FIRST_NAME, LAST_NAME
FROM AUTHOR
WHERE AUTHOR_ID = (
    SELECT MAX(AUTHOR_ID)
    FROM AUTHOR
);
```

```
Task completed in 0.129 seconds

FIRST_NAME                                          LAST_NAME                                           TOTAL_TITLES
--------------------------------------------------  --------------------------------------------------  ------------
J.K.                                                Rowling                                                        1

FIRST_NAME                                          LAST_NAME
--------------------------------------------------  --------------------------------------------------
Agatha                                              Christie
```

5.Data Manipulation (DML)

☐ Include examples of the following commands:

▪INSERT – Add new records to a table

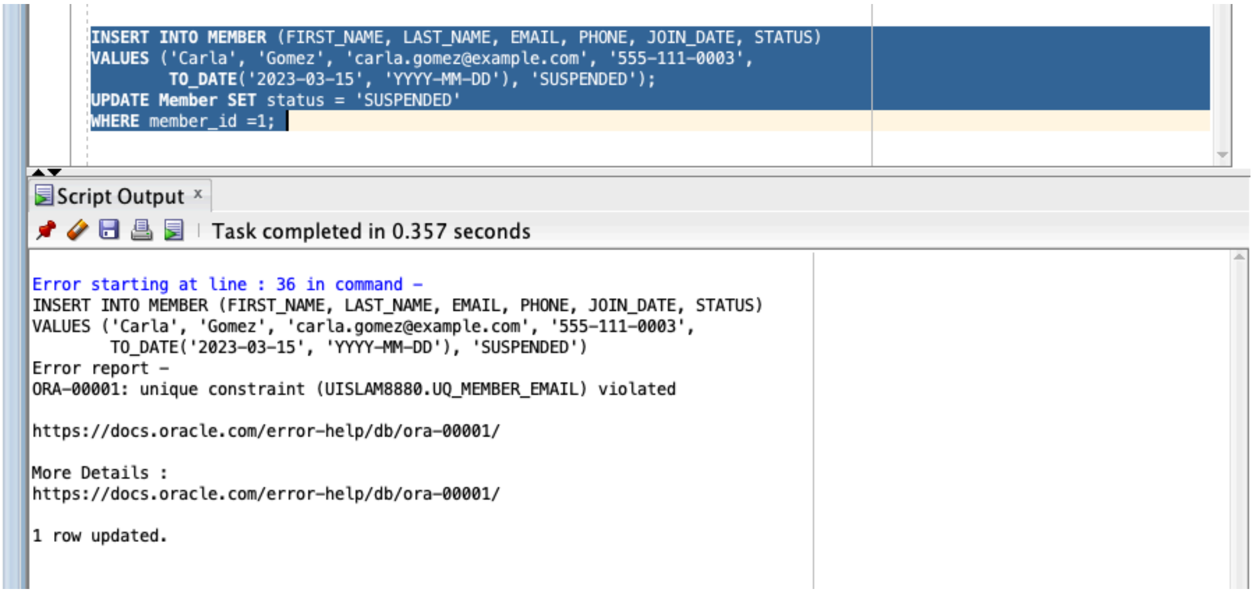
▪UPDATE –Modify existing records

▪DELETE – Remove records from a table

oProvide at least 2 examples of each command type

1. **Using Update command:**

INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Carla', 'Gomez', 'carla.gomez@example.com', '555-111-0003',
    TO_DATE('2023-03-15', 'YYYY-MM-DD'), 'SUSPENDED');
UPDATE Member SET status = 'SUSPENDED'
WHERE member_id =1;



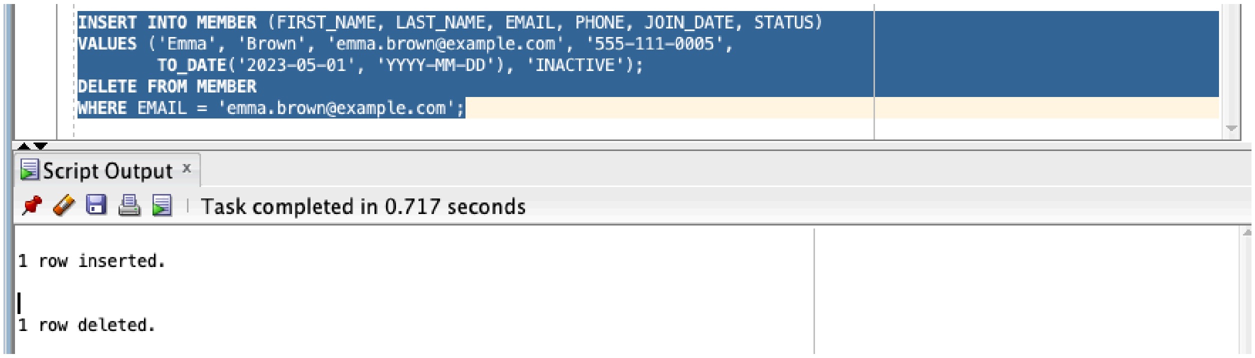
```
INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Carla', 'Gomez', 'carla.gomez@example.com', '555-111-0003',
        TO_DATE('2023-03-15', 'YYYY-MM-DD'), 'SUSPENDED');
UPDATE Member SET status = 'SUSPENDED'
WHERE member_id =1;
```

Task completed in 0.357 seconds

```
Error starting at line : 36 in command -
INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Carla', 'Gomez', 'carla.gomez@example.com', '555-111-0003',
        TO_DATE('2023-03-15', 'YYYY-MM-DD'), 'SUSPENDED')
Error report -
ORA-00001: unique constraint (UISLAM8880.UQ_MEMBER_EMAIL) violated

https://docs.oracle.com/error-help/db/ora-00001/

More Details :
https://docs.oracle.com/error-help/db/ora-00001/

1 row updated.
```

2. **Using DELETE command:**

INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Emma', 'Brown', 'emma.brown@example.com', '555-111-0005',
    TO_DATE('2023-05-01', 'YYYY-MM-DD'), 'INACTIVE');
DELETE FROM MEMBER
WHERE EMAIL = 'emma.brown@example.com';



```
INSERT INTO MEMBER (FIRST_NAME, LAST_NAME, EMAIL, PHONE, JOIN_DATE, STATUS)
VALUES ('Emma', 'Brown', 'emma.brown@example.com', '555-111-0005',
        TO_DATE('2023-05-01', 'YYYY-MM-DD'), 'INACTIVE');
DELETE FROM MEMBER
WHERE EMAIL = 'emma.brown@example.com';
```

Task completed in 0.717 seconds

```
1 row inserted.

1 row deleted.
```

5.Constraints and Business Rules:

Main Business Rules:

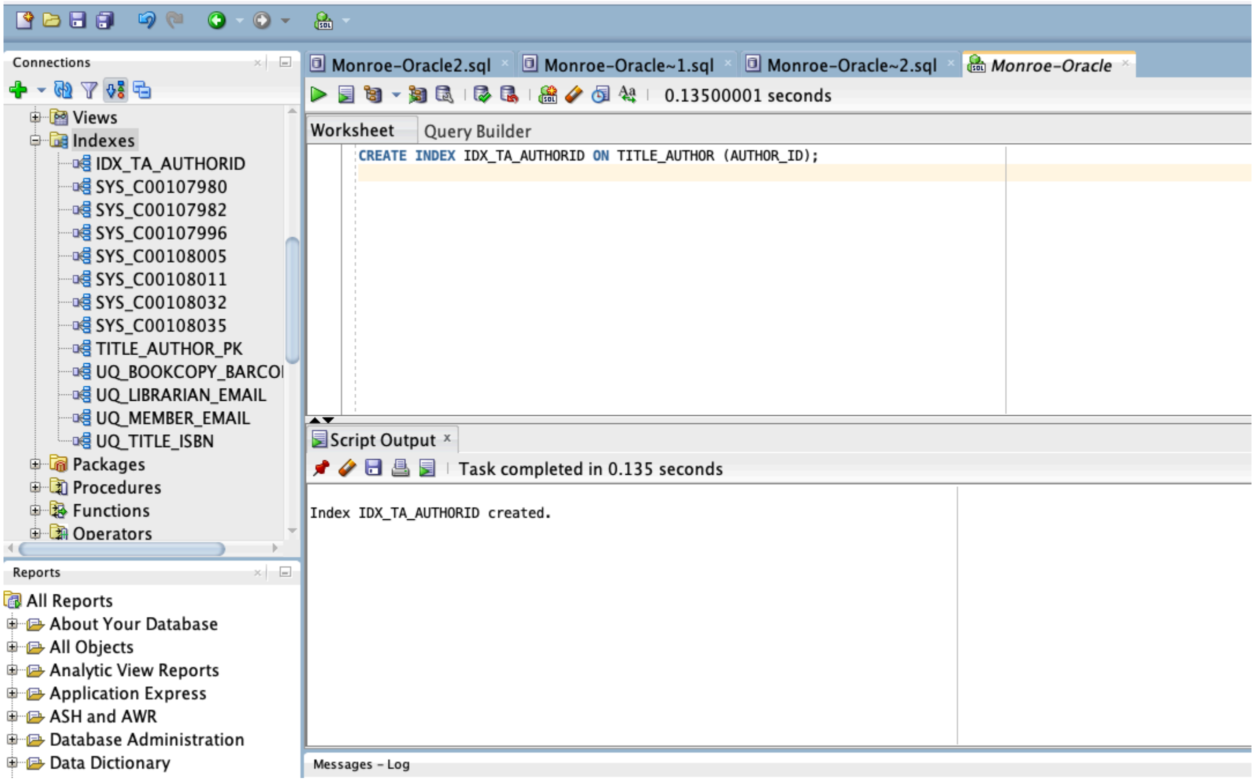
1. Each library member must have a unique record.
2. Each book title may have one or more authors.
3. A title can have multiple physical copies, and each copy must have a unique barcode.
4. Every time a member borrows a book copy, a loan record must be created.
5. A loan must include a due date.
6. If a book is returned late, a fine must be recorded and linked to the loan.
7. Librarians must be tracked because they process checkouts and returns.
8. A book copy cannot be loaned to more than one member at the same time.
9. Authors and titles must be linked because some titles have multiple authors.

How Constraints Enforce the Business Rules:

1. Primary keys guarantee that each book, author, member, or transaction is stored only once and can always be uniquely identified.
2. Unique constraints, such as the one on ISBN, prevent duplicate records for the same book. NOT NULL constraints ensure that important fields—like names and contact information—are always provided and cannot be left empty.
3. Foreign keys maintain the correct relationships between tables by ensuring that linked records actually exist; for example, an author cannot be assigned to a book unless both are already in the system. Composite keys in relationship tables, like TITLE_AUTHOR, prevent duplicate pairings and keep many-to-many relationships clean.

6. Performance and Optimization:

1. One index with Foreign Key:

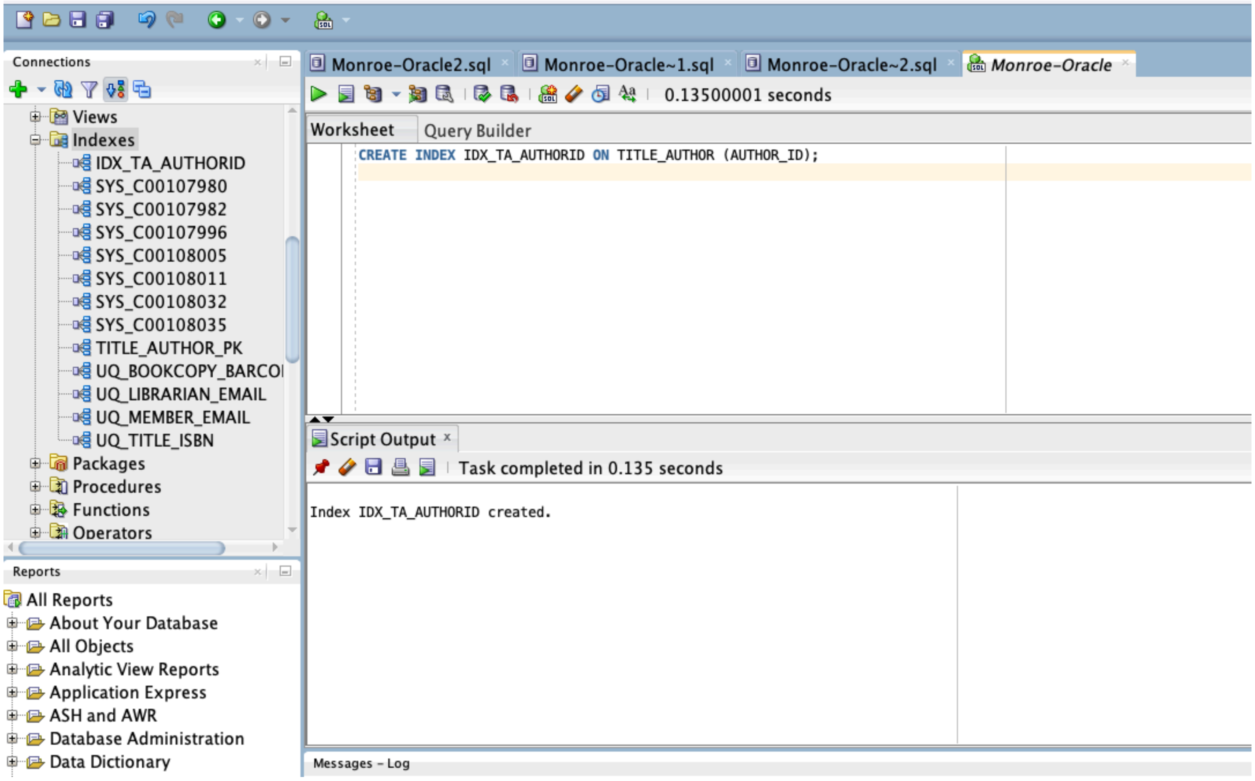


```
CREATE INDEX IDX_TA_AUTHORID ON TITLE_AUTHOR (AUTHOR_ID);
```

```
Index IDX_TA_AUTHORID created.
```

2. Index performance before & after:

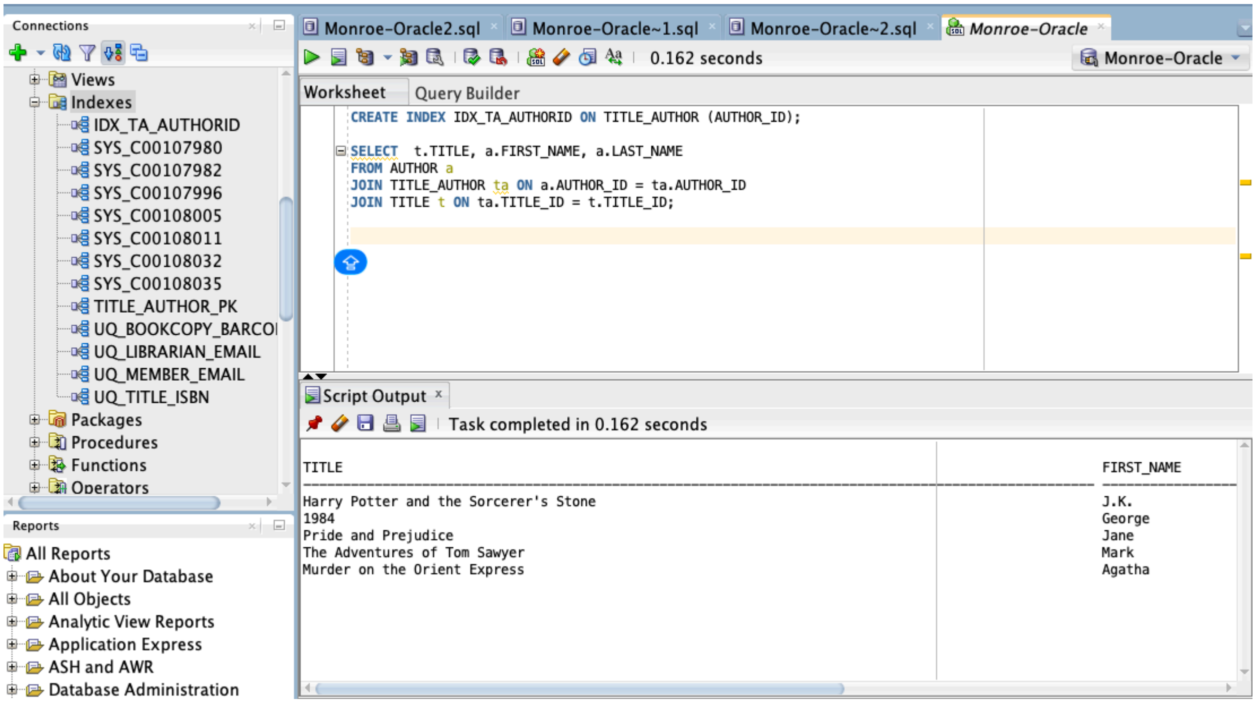
Before: Creating the Index



```
CREATE INDEX IDX_TA_AUTHORID ON TITLE_AUTHOR (AUTHOR_ID);
```

```
Index IDX_TA_AUTHORID created.
```

After: Creating the Index



```
CREATE INDEX IDX_TA_AUTHORID ON TITLE_AUTHOR (AUTHOR_ID);

SELECT  t.TITLE, a.FIRST_NAME, a.LAST_NAME
FROM AUTHOR a
JOIN TITLE_AUTHOR ta ON a.AUTHOR_ID = ta.AUTHOR_ID
JOIN TITLE t ON ta.TITLE_ID = t.TITLE_ID;
```

```
TITLE                                                                                   FIRST_NAME
----------------------------------------------------------------------------------------------------
Harry Potter and the Sorcerer's Stone                                                   J.K.
1984                                                                                    George
Pride and Prejudice                                                                     Jane
The Adventures of Tom Sawyer                                                            Mark
Murder on the Orient Express                                                            Agatha
```

Oracle uses the index to find matching AUTHOR_ID values much faster. Instead of reading the whole TITLE_AUTHOR table, it performs an indexing, jumping directly to the rows it needs.

Conceptually if performance difference is not visible: Without the index slow down the insert/update performance. With index improves data retrieval speed.

3. An index is a special data structure (often a B-tree) that allows faster data retrieval without scanning the entire table. It stores key values with pointers to the corresponding rows in sorted order. Query Optimization means Oracle refers to all the possible ways to run a query and picks the fastest one. Hashing uses a mathematical function to compute the physical location of a record. It’s very efficient for equality searches (e.g., finding a record with a specific ID). However, it’s not ideal for range based queries (like >,< , or BETWEEN).

Reflection

Throughout this project, I reflect on understanding of how to design, create and implement Logical database, Relational database using Oracle SQL developer and SQL data modeler. I also learned how to use a luchi chart to create diagrams imported to SQL developer apps. I learned how to download and create accounts in SQL developer apps each time I needed to create new connections. I also learned how to use real-world scenarios into a logical data model and then convert that into relational schema using SQL data modeler. From creating tables, defining the relationships and how to apply the constraints made me realize how data integrity is maintained in a well structured database. I also created SQL script through SQL worksheet including the SELECT, SELECT, FROM,WHERE,JOIN,INNER JOIN, AGGREGATION FUNCTION which improved my confidence in retrieving and analyzing the data efficiently. Building and testing indexes also gave me confidence how performance optimization works and how indexing can help to speed up the queries that rely on the frequently searched columns. One of the biggest challenges faced was to create the SQL queries in order to create the tables. At times I did connect to my professor to walk me through which made it easier when he provided me sample data to run and create the tables. However, this project improved my ability to adapt and learn from the process regardless how hard it was. Using SQL data modular helped visualize the logical and relational database more clearly, while Oracle SQL developer gave me hands-on experience of implementing the schema, run queries and create indexes. However, this experience gave me the opportunity to learn something new and apply in real world business scenarios to apply in future database driven applications.