```
#!/usr/bin/env python3
### Assignment 3 - SQL statements
### Author: Jeremiah Purba

#imports at top of file
import sqlite3
import logging
import os.path

# For logging
def debug_config():
      logging.basicConfig(
        level=logging.DEBUG,
        format = "[Movies]:%(asctime)s:%(levelname)s:%(message)s"
    )  #DEBUG,INFO,ERROR,WARNING,CRITICAL

# For checking file
def db_checkfile(dbfile):
    # get current directory
    path = os.getcwd()
    db_path = os.path.join(path, dbfile)
    if os.path.exists(dbfile) and os.path.getsize(dbfile) > 0:
        logging.debug("{} found and not zero size".format(db_path))
    else:
        logging.error("{a} not found or zero size".format(a=dbfile))

# For connecting to DB
def db_connect(dbfile):
      con = sqlite3.connect(dbfile)
      logging.debug("DB Connected".format())
      return con

# For cursor
def db_cursor(con):
      cur = con.cursor()
      logging.debug("Cursor set".format())
      return cur


#print program title
def display_title():
    print("My Movie Database")
    print()

def create_table():
    dbfile = "movies_purba.db"

    db_checkfile(dbfile)

    # Connect to DB
    con = db_connect(dbfile)

    # Get the cursor
    cur = db_cursor(con)

    # query to create the table1
    # to avoid duplpicaton use UNIQUE
    create_table_query1 = '''
    CREATE TABLE IF NOT EXISTS movies_info_1 (
```

```
    show_id INTEGER NOT NULL PRIMARY KEY,
    genre TEXT NOT NULL,
    title TEXT NOT NULL,
    director TEXT NOT NULL,
    UNIQUE (show_id,genre,title,director)
)'''

# Attempt to create table1
try:
    cur.execute(create_table_query1)
    logging.debug("Create movies_info_1 table ".format())
except sqlite3.IntegrityError as e:
     print(f"First table creation IntegrityError: {e}")


# query to create the table2
# to avoid duplpicaton use UNIQUE
create_table_query2 = '''
CREATE TABLE IF NOT EXISTS movies_info_2 (
    show_id INTEGER NOT NULL,
    release_year INTEGER NOT NULL,
    description TEXT NOT NULL,
    UNIQUE (show_id,release_year,description)
    FOREIGN KEY (show_id) REFERENCES movies_info_1(show_id)
)'''

# Attempt to create table1
try:
    cur.execute(create_table_query2)
    logging.debug("Create movies_info_2 table ".format())
except sqlite3.IntegrityError as e:
     print(f"Second table creation IntegrityError: {e}")


# query to insert data into the table1:
insert_table1_query = """
INSERT OR REPLACE INTO movies_info_1 (show_id,genre,title,director)
    VALUES (?, ?, ?, ?)
"""
data_table1 = [
(1,'Animation','Toy Story', 'Stanton'),
(2,'Animation','Finding Nemo', 'Stanton'),
(3,'Animation','Cars', 'Lasseter')
]
cur.executemany(insert_table1_query, data_table1)


# query to insert data into table2:
insert_table2_query = """
INSERT OR REPLACE INTO movies_info_2 (show_id,release_year,description)
    VALUES (?, ?, ?)
"""
data_table2 = [
(1,1995,'Stars come to life as they work to be reunited with Andy'),
(2,2003,'Adventures of Nemo and his friend, Dory'),
(3,2006,'Story of a car lost in Radiator Springs')
]

cur.executemany(insert_table2_query, data_table2)
```

```
    # Print the table header
    header1_query = "SELECT * FROM movies_info_1"
    header2_query = "SELECT * FROM movies_info_2"

    col_Header1 = cur.execute(header1_query)

    #print the tables headers

    table1_header = []
    for column in col_Header1.description:
        header = column[0]
        table1_header.append(header)

    col_Header2 = cur.execute(header2_query)
    table2_header = []
    for column2 in col_Header2.description:
        header2 = column2[0]
        table2_header.append(header2)

    print(table1_header[1], " ", table1_header[2], " ",table1_header[3],
    " ", table2_header[1], " ", table2_header[2]
    )


    join_query = """
    SELECT movies_info_1.*, movies_info_2.*
    FROM movies_info_1 INNER JOIN movies_info_2
    ON movies_info_1.show_id = movies_info_2.show_id
    """

    cur.execute(join_query)
    join_results = cur.fetchall()

    for row in join_results:
        print(row[1], "  ", row[2], "  ", row[3],
        "  ", row[5], "  ", row[6])

    con.commit()
    if con:
        con.close()
        logging.debug("DB Closed")


def main():
    #display the program title
    display_title()
    # logging config
    debug_config()
    # Create table
    create_table()

    print("All done!")

if __name__ == "__main__":
    main()
```