UOW MALAYSIA KDU PENANG UNIVERSITY COLLEGE

PART OF THE UNIVERSITY OF WOLLONGONG AUSTRALIA GLOBAL NETWORK

Programme	**Course Code and Title**
Diploma in Computer Studies/Information Technology	DDS1144 Database Systems
Students Name & ID	**Lecturer Name:** Danny Chen
1. 0204677 LIM ZHE YUAN 2. 0205096 THOR WEN ZHENG 3. 0204274 TAN YEU CHEN	

Date of Assignment Release	**Submission Deadline**	**Indicative Weighting**
16/07/2021	11.59pm, Friday, 13/08/2021	100 Marks, Weighted @ 15%

Assignment title	Assignment 02 – The GMG Movie Database Implementation

Student's declaration

I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.

Students signatures: Date: 12 August 2021

1. Zhe Yuan
2. ThoR
3. Yeu Chen

Plagiarism

The assignment is based on an individual response. The report must be **completely your own work** and you must not copy from others. Any plagiarized work will be zero-rated. Any reference material you use (books, journals, Internet, magazines etc.) must be clearly identified in your report using procedures in the Harvard System of Referencing.

TABLE OF CONTENTS

ASSIGNMENT 02 – THE GMG MOVIE DATABASE IMPLEMENTATION

(Max Marks: 100, Weighted @ 15% of Total Course Marks)

Your database development team (2-3 members) have completed designing the database for GMG and now have to implement the database system by creating the tables based on your design from assignment 1.

TASK REQUIREMENTS:

1. Generate the SQL statements to create tables defined by the ER-diagram and data dictionary you created in assignment 1. Include any possible constraints that will affect each table, the storage parameters for each table, the relationships between tables.
   (Note: pay attention to the order in which the tables will need to be created, ensuring referential integrity throughout the database.)

   (60 marks)

2. For each of the table you created in task 1, generate an SQL statement to insert a record containing appropriate data of your choosing for the table.
   (Note: pay attention to the order in which the records will need to be inserted, ensuring referential integrity throughout the database.)

   (24 marks)

3. Generate SQL statements to retrieve data for the following tasks:
   a. List all the movies currently showing using the following output format example:
      You can watch **Mission Impossible: Fallout**, rated **PG13**, starring **Tom Cruise** at **GSC Gurney Plaza**.
   b. Give the name of movies that have sold less tickets than the average number of tickets sold.
   c. List the Top 10 highest paid actors in the database.
   d. List the Top 10 movies of all times by ticket receipts.

   (16 marks)

REPORT FORMAT:

- All solutions must be typed using MSWord.
- Font type: Cambria (headers), Calibri (body)
- Font Size: 12/14 (headers), 11 (body)
- Line spacing is 1.15

TASK 1: CREATING TABLES

```
CREATE TABLE Producer (
        producer_id          VARCHAR(6),
        producer_firstname   VARCHAR(50) NOT NULL,
        producer_lastname    VARCHAR(50) NOT NULL,
        producer_gender      CHAR(1) DEFAULT 'O' CONSTRAINT chk_producer_gender CHECK
(producer_gender IN ('M', 'F', 'O')),
       CONSTRAINT pk_producer PRIMARY KEY (producer_id)
)
STORAGE (
INITIAL 430K
NEXT 43K
PCTINCREASE 0
);


CREATE TABLE Movie (
        movie_id         VARCHAR(6),
        movie_name       VARCHAR(100) NOT NULL,
        year_produced    NUMBER(4) CONSTRAINT chk_year CHECK (year_produced > 0),
        genre            VARCHAR(15),
        runtime          NUMBER(3) NOT NULL,
        rating           VARCHAR(5) DEFAULT 'U' CONSTRAINT chk_rating CHECK (rating IN ('G',
'PG', 'PG-13', 'R', 'NC-18', 'U', 'P13', '18')),
        producer_id      VARCHAR(6),
        CONSTRAINT pk_movie PRIMARY KEY (movie_id),
        CONSTRAINT fk_movie_producer FOREIGN KEY (producer_id) REFERENCES Producer
(producer_id)
)
STORAGE (
INITIAL 2M
NEXT 150K
PCTINCREASE 0
);
```

```
CREATE TABLE Actor (
	actor_id		VARCHAR(6),
	actor_firstname VARCHAR(50) NOT NULL,
	actor_lastname  VARCHAR(50) NOT NULL,
	actor_gender	CHAR(1) DEFAULT 'O' CONSTRAINT chk_actor_gender CHECK (actor_gender
IN ('M', 'F', 'O')),
	CONSTRAINT pk_actor PRIMARY KEY (actor_id)
)
STORAGE (
INITIAL 100K
NEXT 20K
PCTINCREASE 0
);


CREATE TABLE CastMember (
	movie_id		VARCHAR(6),
	actor_id		VARCHAR(6),
	role			CHAR(1) DEFAULT 'S' CONSTRAINT chk_role CHECK (role IN ('M', 'S')),
	payment		NUMBER(9) CONSTRAINT chk_payment CHECK (payment > 0),
	CONSTRAINT pk_castmember PRIMARY KEY (movie_id, actor_id),
	CONSTRAINT fk_castmember_movie FOREIGN KEY (movie_id) REFERENCES Movie
(movie_id),
	CONSTRAINT fk_castmember_actor FOREIGN KEY (actor_id) REFERENCES Actor (actor_id)
)
STORAGE (
INITIAL 2M
NEXT 140K
PCTINCREASE 0
);
```

```
CREATE TABLE Theatre (
        theatre_id              VARCHAR(6),
        theatre_name            VARCHAR(30) NOT NULL,
        theatre_status          CHAR(1) DEFAULT 'U' CONSTRAINT chk_status CHECK
(theatre_status IN ('Y', 'N', 'U')),
        price_per_ticket        NUMBER CONSTRAINT chk_price CHECK (price_per_ticket > 0),
        address_unit            VARCHAR(10) NOT NULL,
        street                  VARCHAR(50) NOT NULL,
        city                    VARCHAR(30) NOT NULL,
        postcode                VARCHAR(10) NOT NULL,
        state                   VARCHAR(30) NOT NULL,
        country                 VARCHAR(30) NOT NULL,
        CONSTRAINT pk_theatre PRIMARY KEY (theatre_id)
)
STORAGE (
INITIAL 220K
NEXT 22K
PCTINCREASE 0
);


CREATE TABLE TicketSale (
        movie_id        VARCHAR(6),
        theatre_id      VARCHAR(6),
        quantity_sold   NUMBER NOT NULL,
        return_sales    NUMBER NOT NULL,
        CONSTRAINT pk_ticketsale PRIMARY KEY (movie_id, theatre_id),
        CONSTRAINT fk_ticketsale_movie FOREIGN KEY (movie_id) REFERENCES Movie (movie_id),
        CONSTRAINT fk_ticketsale_theatre FOREIGN KEY (theatre_id) REFERENCES Theatre
(theatre_id)
)
STORAGE (
INITIAL 10M
NEXT 1M
PCTINCREASE 0
);
```

TASK 2: INSERTING A RECORD INTO EACH TABLE

```
INSERT INTO Producer
VALUES ('P00001', 'George', 'Lucas', 'M');

INSERT INTO Actor
VALUES ('A00001', 'Mark', 'Hamill', 'M');

INSERT INTO Movie
VALUES ('M00001', 'Star Wars: Episode VI - Return of the Jedi', 1983, 'SciFi',  132, 'PG', 'P00001');

INSERT INTO CastMember
VALUES ('M00001', 'A00001', 'M', 500000);

INSERT INTO Theatre
VALUES ('T00001', 'GSC Gurney Plaza', 'Y', 15, '170-07-01', 'Gurney Drive, Pulau Tikus', 'George
Town', '10250', 'Penang', 'Malaysia');

INSERT INTO TicketSale
VALUES ('M00001', 'T00001', 10000, 150000);

COMMIT;
```

TASK 3: RETRIEVING DATA

a.
```
SELECT 'You can watch ' || movie_name || ', rated ' || rating || ', starring ' || actor_firstname ||
' ' || actor_lastname || ' at ' || theatre_name || '.' AS "All movies currently showing"
FROM Movie INNER JOIN CastMember USING (movie_id)
INNER JOIN Actor USING (actor_id)
INNER JOIN TicketSale USING (movie_id)
INNER JOIN Theatre USING (theatre_id)
WHERE role = 'M'
ORDER BY theatre_id;
```

b.
```
SELECT movie_name
FROM Movie INNER JOIN TicketSale USING (movie_id)
GROUP BY movie_name
HAVING SUM(quantity_sold) < (SELECT AVG(quantity_sold) FROM TicketSale);
```

c.
```
SELECT Actor.actor_id, actor_firstname || ' ' || actor_lastname AS "actor_name",
SUM(payment) AS "total_payment"
FROM Actor INNER JOIN CastMember ON (Actor.actor_id = CastMember.actor_id)
GROUP BY Actor.actor_id, actor_firstname, actor_lastname
ORDER BY SUM(payment) DESC
FETCH FIRST 10 ROWS ONLY;
```

d.
```
SELECT Movie.movie_id, movie_name, SUM(quantity_sold) AS "total_tickets_sold"
FROM Movie INNER JOIN TicketSale ON (Movie.movie_id = TicketSale.movie_id)
GROUP BY Movie.movie_id, movie_name
ORDER BY SUM(quantity_sold) DESC
FETCH FIRST 10 ROWS ONLY;
```

MARKING RUBRIC

DDS1144 Database Systems

Assignment 2 – The GMG Movie Database Implementation (100 Marks, Weighted @ 15%)

Student ID: 0204677 ***Student Name: LIM ZHE YUAN***
Student ID: 0205096 ***Student Name: THOR WEN ZHENG***
Student ID: 0204274 ***Student Name: TAN YEU CHEN***

LEARNING OUTCOME	MARKING CRITERIA	SCALE					Task Marks (Max. 100)	Task Weighted Marks
		Fail (0-49)	**3rd Class (50-59)**	**2nd Lower Class (60-69)**	**2nd Upper Class (70-79)**	**1st Class (80-100)**		
CLO4: Create and manipulate relations in databases by using Database Definition Language (DDL) and Database Manipulation Language	**Task 1: Generate the SQL statements to create tables (60 marks)**	*Generated little-to-no required statements to create tables and insert records accurately and completely.*	*Generated few-to-some of the required statements to create tables and insert records accurately and completely.*	*Generated some-to-most of the required statements to create tables and insert records accurately and completely.*	*Generated most-to-almost-all of the required statements to create tables and insert records accurately and completely.*	*Generated all of the required statements to create tables and insert records accurately and completely.*		/60
	Task 2: Generate SQL statements to insert data (24 marks)	*Generated little-to-no required statements to create tables and insert records accurately and completely.*	*Generated few-to-some of the required statements to create tables and insert records accurately and completely.*	*Generated some-to-most of the required statements to create tables and insert records accurately and completely.*	*Generated most-to-almost-all of the required statements to create tables and insert records accurately and completely.*	*Generated all of the required statements to create tables and insert records accurately and completely.*		/24
	Task 3: Generate SQL statements to retrieve data (16 marks)	*Generated little-to-no required statements to retrieve data/records accurately and completely.*	*Generated few-to-some of the required statements to retrieve data/records accurately and completely.*	*Generated some-to-most of the required statements to retrieve data/records accurately and completely.*	*Generated most-to-almost-all of the required statements to retrieve data/records accurately and completely.*	*Generated all of the required statements to retrieve data/records accurately and completely.*		/16
							Total (100%)	/100
							Weighted Marks @ 15%	/15