FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGIES

FALL 2023

CA EXAMINATION

NAMES: NDE HURICH DILAN

MATRICULE:ICTU20223351

EMAIL: nde.dilan@ictuniversity.edu.cm

TEL:+237694525931

RELATIONAL DATABASE CA

Exercice 1: CINEMA

First let's create the database we will be working with, and fill it with some dummy data:

```
mysql> CREATE DATABASE CINEMA;
Query OK, 1 row affected (0.01 sec)

mysql> use CINEMA;
Database changed
```

Here we are just creating the FILM TABLE inside the CINEMADB

```
mysql> use CINEMA;
Database changed
mysql> CREATE TABLE FILM (
    ->     FILM_NUMBER INT PRIMARY KEY,
    ->     TITLE VARCHAR(255) NOT NULL,
    ->     TYPE VARCHAR(50),
    ->     YEAR INT,
    ->     DURATION INT,
    ->     BUDGET DECIMAL(10, 2),
    ->     DIRECTOR VARCHAR(100),
    ->     SALARY DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
```

Now we insert some values in the film table(anime movie)

```
mysql> INSERT INTO FILM (FILM_NUMBER, TITLE, TYPE, YEAR, DURATION, BUDGE
T, DIRECTOR, SALARY)
    -> VALUES
    ->   (1, 'Spirited Away', 'Anime', 2001, 125, 15000000, 'Hayao Miyaz
aki', 500000),
    ->   (2, 'Your Name', 'Anime', 2016, 107, 3800000, 'Makoto Shinkai',
 450000),
    ->   (3, 'Attack on Titan: Crimson Bow and Arrow', 'Anime Movie', 20
14, 119, 12000000, 'Tetsurō Araki', 550000),
```

I) Find the list of all the films.

```
mysql> SELECT * FROM FILM;
+-------------+-----------------------------------------------+-------------+------+----------+-------------+--------------------+-----------+
| FILM_NUMBER | TITLE                                         | TYPE        | YEAR | DURATION | BUDGET      | DIRECTOR           | SALARY    |
+-------------+-----------------------------------------------+-------------+------+----------+-------------+--------------------+-----------+
|           1 | Spirited Away                                 | Anime       | 2001 |      125 | 15000000.00 | Hayao Miyazaki     | 500000.00 |
|           2 | Your Name                                     | Anime       | 2016 |      107 |  3800000.00 | Makoto Shinkai     | 450000.00 |
|           3 | Attack on Titan: Crimson Bow and Arrow        | Anime Movie | 2014 |      119 | 12000000.00 | Tetsur? Araki      | 550000.00 |
|           4 | Demon Slayer: Mugen Train                     | Anime Movie | 2020 |      117 | 10000000.00 | Haruo Sotozaki     | 600000.00 |
|           5 | One Punch Man: Road to Hero                   | Anime Movie | 2015 |       65 |  5000000.00 | Shingo Natsume     | 400000.00 |
|           6 | Neon Genesis Evangelion: The End of Evangelion | Anime Movie | 1997 |       87 |  8000000.00 | Hideaki Anno       | 600000.00 |
|           7 | Cowboy Bebop: The Movie                       | Anime Movie | 2001 |      115 |  9000000.00 | Shinichir? Watanabe | 550000.00 |
|           8 | Princess Mononoke                             | Anime       | 1997 |      134 | 23000000.00 | Hayao Miyazaki     | 550000.00 |
|           9 | Weathering with You                           | Anime       | 2019 |      112 | 11000000.00 | Makoto Shinkai     | 500000.00 |
|          10 | Sword Art Online: Ordinal Scale               | Anime Movie | 2017 |      119 |  7000000.00 | Tomohiko It?       | 450000.00 |
|          11 | Akira                                         | Anime Movie | 1988 |      124 |  9000000.00 | Katsuhiro Otomo    | 500000.00 |
|          12 | My Neighbor Totoro                            | Anime       | 1988 |       86 |  3700000.00 | Hayao Miyazaki     | 450000.00 |
|          13 | Ghost in the Shell                            | Anime Movie | 1995 |       83 | 10000000.00 | Mamoru Oshii       | 550000.00 |
|          14 | Dragon Ball Super: Broly                      | Anime Movie | 2018 |      101 |  8500000.00 | Tatsuya Nagamine   | 600000.00 |
|          15 | Grave of the Fireflies                        | Anime       | 1988 |       89 |  3500000.00 | Isao Takahata      | 450000.00 |
|          16 | One Piece: Stampede                           | Anime Movie | 2019 |      101 | 12000000.00 | Takashi Otsuka     | 550000.00 |
+-------------+-----------------------------------------------+-------------+------+----------+-------------+--------------------+-----------+
16 rows in set (0.00 sec)

mysql>
```

According to the previous table no film exceeds 180 mins

ii) Find the list of films whose length exceeds 180 minutes

```
mysql> SELECT FILM_NUMBER FROM FILM WHERE DURATION > 180;
Empty set (0.00 sec)

mysql>
```

ii) Give the list of all the type of films

```
mysql> SELECT DISTINCT TYPE FROM FILM;
+---------------+
| TYPE          |
+---------------+
| Anime         |
| Anime Movie   |
| Drama movie   |
| Fantasy Movie |
| Action Movie  |
+---------------+
5 rows in set (0.12 sec)

mysql>
```

The different type of film present inside the FILM TABLE

iv) Give the number of films by type

```
mysql> SELECT TYPE , COUNT(TYPE) AS NUMBER_OF_FILMS FROM FILM GROUP BY TYPE;
+---------------+-----------------+
| TYPE          | NUMBER_OF_FILMS |
+---------------+-----------------+
| Anime         |               6 |
| Anime Movie   |              10 |
| Drama movie   |               1 |
| Fantasy Movie |               1 |
| Action Movie  |               1 |
+---------------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

v) Find the title(s) and year(s) of the longest film(s).

```
mysql> SELECT TITLE,YEAR FROM FILM ORDER BY DURATION DESC LIMIT 1;
+-------------------+------+
| TITLE             | YEAR |
+-------------------+------+
| Princess Mononoke | 1997 |
+-------------------+------+
1 row in set (0.00 sec)

mysql>
```

I have to say that here I was confused because of the 's' at the end, but I choose to limit the result to **1**.

vi) Find all the "pairs of actors", i.e., the actors having played the "Lead" role in the same film.

```
mysql> CREATE TABLE DISTRIBUTION(FILM_NUMBER INT , ACTOR_NUMBER INT, ROLE VARCHAR(45), SALARY INT);
Query OK, 0 rows affected (0.17 sec)

mysql> INSERT INTO DISTRIBUTION (FILM_NUMBER, ACTOR_NUMBER, ROLE, SALARY)
    -> VALUES
    ->   (1, 1, 'Chihiro', 2000000),
    ->   (1, 2, 'No-Face', 1500000),
    ->   (2, 3, 'Satsuki', 1800000),
    ->   (2, 4, 'Totoro', 1600000),
    ->   (3, 5, 'Kaneda', 2500000),
    ->   (3, 6, 'Tetsuo', 2000000);
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql>
```

Let's create and fill the **DISTRIBUTION** table

Now we can perform operation on the **DISTRIBUTION** table.

```
mysql> CREATE TABLE PERSON(PERSON_NUMBER INT, LASTNAME VARCHAR(45) , FIRSTNAME VARCHAR(45) );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO PERSON (PERSON_NUMBER, LASTNAME, FIRSTNAME)
    -> VALUES
    ->   (1, 'Kidman', 'Nicole'),
    ->   (2, 'Smith', 'Will'),
    ->   (3, 'Blanchett', 'Cate'),
    ->   (4, 'Hanks', 'Tom'),
    ->   (5, 'Damon', 'Matt'),
    ->   (6, 'Portman', 'Natalie');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql>
```

```
mysql> CREATE TABLE ACTOR(ACTOR_NUMBER INT PRIMARY KEY, AGENT VARCHAR(45), SPECIALITY VARCHAR(45), HEIGHT INT, WEIGHT INT);
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO ACTOR (ACTOR_NUMBER, AGENT, SPECIALITY, HEIGHT, WEIGHT)
    -> VALUES
    ->   (1, 'Agent1', 'Drama', 175, 65),
    ->   (2, 'Agent2', 'Fantasy', 180, 70),
    ->   (3, 'Agent3', 'Adventure', 165, 55),
    ->   (4, 'Agent4', 'Family', 175, 68),
    ->   (5, 'Agent5', 'Sci-Fi', 178, 72),
    ->   (6, 'Agent6', 'Action', 160, 50);
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql>
```

```
mysql> SELECT * FROM ACTOR;
+--------------+--------+------------+--------+--------+
| ACTOR_NUMBER | AGENT  | SPECIALITY | HEIGHT | WEIGHT |
+--------------+--------+------------+--------+--------+
|            1 | Agent1 | Drama      |    175 |     65 |
|            2 | Agent2 | Fantasy    |    180 |     70 |
|            3 | Agent3 | Adventure  |    165 |     55 |
|            4 | Agent4 | Family     |    175 |     68 |
|            5 | Agent5 | Sci-Fi     |    178 |     72 |
|            6 | Agent6 | Action     |    160 |     50 |
+--------------+--------+------------+--------+--------+
6 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM PERSON;
+---------------+-----------+-----------+
| PERSON_NUMBER | LASTNAME  | FIRSTNAME |
+---------------+-----------+-----------+
|             1 | Kidman    | Nicole    |
|             2 | Smith     | Will      |
|             3 | Blanchett | Cate      |
|             4 | Hanks     | Tom       |
|             5 | Damon     | Matt      |
|             6 | Portman   | Natalie   |
+---------------+-----------+-----------+
6 rows in set (0.00 sec)
```

vii) Find the names of people who are not agents, actors or directors.

```
mysql> SELECT * FROM PERSON;
+---------------+----------+----------+
| PERSON_NUMBER | LASTNAME | FIRSTNAME |
+---------------+----------+----------+
|             1 | Kidman   | Nicole   |
|             2 | Smith    | Will     |
|             3 | Blanchett | Cate    |
|             4 | Hanks    | Tom      |
|             5 | Damon    | Matt     |
|             6 | Portman  | Natalie  |
|             7 | Man      | Cold     |
|             8 | Eng      | Maxkit   |
|             9 | NDE      | Dilan    |
+---------------+----------+----------+
9 rows in set (0.00 sec)

mysql>
```

Here is the PERSON table, I've added some persons which are not actors (like myself) in the table.

Now we can perform operation on the **PERSON** table.

```
mysql> SELECT FIRSTNAME, LASTNAME FROM PERSON WHERE PERSON_NUMBER NOT IN ( SELECT ACTOR_NUMBER FROM ACTOR UNION SELECT AGENT FROM ACTOR UNION SELECT
 DIRECTOR FROM FILM);
+-----------+----------+
| FIRSTNAME | LASTNAME |
+-----------+----------+
| Cold      | Man      |
| Maxkit    | Eng      |
| Dilan     | NDE      |
+-----------+----------+
3 rows in set, 75 warnings (0.01 sec)

mysql>
```

From the result we can see my name (the last 😊)

viii) Give the last name and first name of the directors who have acted in at least one of their own films

```
mysql> SELECT P.LASTNAME,P.FIRSTNAME FROM PERSON P JOIN FILM F ON P.PERSON_NUMBER = F.DIRECTOR JOIN DI
STRIBUTION D ON F.FILM_NUMBER = D.FILM_NUMBER AND D.ACTOR_NUMBER = DIRECTOR;
Empty set (0.00 sec)

mysql>
```

After that I ran the command to feed my data.

From our previous tables we can understand why it returns an empty set.

ix) What is the total salary of the actors of the film "The White House"

for this query , I will create 5 actors of that particular film ,a distribution for each actor and the corresponding film which is "The White House";

```
mysql> select * from actor;
+--------------+--------+------------+--------+--------+
| ACTOR_NUMBER | AGENT  | SPECIALITY | HEIGHT | WEIGHT |
+--------------+--------+------------+--------+--------+
|            1 | Agent1 | Drama      |    175 |     65 |
|            2 | Agent2 | Fantasy    |    180 |     70 |
|            3 | Agent3 | Adventure  |    165 |     55 |
|            4 | Agent4 | Family     |    175 |     68 |
|            5 | Agent5 | Sci-Fi     |    178 |     72 |
|            6 | Agent6 | Action     |    160 |     50 |
|          201 | Agent1 | Action     |    175 |     70 |
|          202 | Agent2 | Drama      |    180 |     65 |
|          203 | Agent3 | Comedy     |    165 |     75 |
+--------------+--------+------------+--------+--------+
```

```
mysql> select * from distribution;
+-------------+---------------+-------------+----------+
| FILM_NUMBER | ACTOR_NUMBER  | ROLE        | SALARY   |
+-------------+---------------+-------------+----------+
|           1 |             1 | Chihiro     | 2000000 |
|           1 |             2 | No-Face     | 1500000 |
|           2 |             3 | Satsuki     | 1800000 |
|           2 |             4 | Totoro      | 1600000 |
|           3 |             5 | Kaneda      | 2500000 |
|           3 |             6 | Tetsuo      | 2000000 |
|           1 |             1 | Lead        |  100000 |
|           1 |             2 | Supporting  |   80000 |
|           2 |             2 | Lead        |   90000 |
|           2 |             3 | Lead        |   95000 |
|           3 |             1 | Lead        |  110000 |
|           3 |             3 | Supporting  |   85000 |
|           1 |           201 | Lead        |   15000 |
|           2 |           202 | Supporting  |   10000 |
|           3 |           203 | Lead        |   12000 |
|          20 |           201 | Lead        |   15000 |
|          21 |           202 | Supporting  |   10000 |
|          22 |           203 | Lead        |   12000 |
|          20 |           201 | Lead        |   15000 |
|          20 |           202 | Supporting  |   10000 |
|          20 |           203 | Supporting  |    8000 |
|          20 |           204 | Supporting  |    9000 |
|          20 |           205 | Supporting  |    8500 |
+-------------+---------------+-------------+----------+
23 rows in set (0.00 sec)

mysql> |
```

```
|          20 | The White House                                | Action       | 2023 |      120 |   500000.00 | 101                |  10000.00 |
```

Here is the query

```
SELECT SUM(SALARY) AS TotalSalary
```

```
FROM DISTRIBUTION D

JOIN FILM F ON D.FILM_NUMBER = F.FILM_NUMBER WHERE F.TITLE = 'The White House';
```

```
mysql> SELECT SUM(D.SALARY) AS TotalSalary FROM DISTRIBUTION D JOIN FILM F ON D.FILM_NUMBER = F.FILM_NUMBER WHERE F.TITLE = "The White House";
+-------------+
| TotalSalary |
+-------------+
|       65500 |
+-------------+
1 row in set (0.01 sec)

mysql>
```

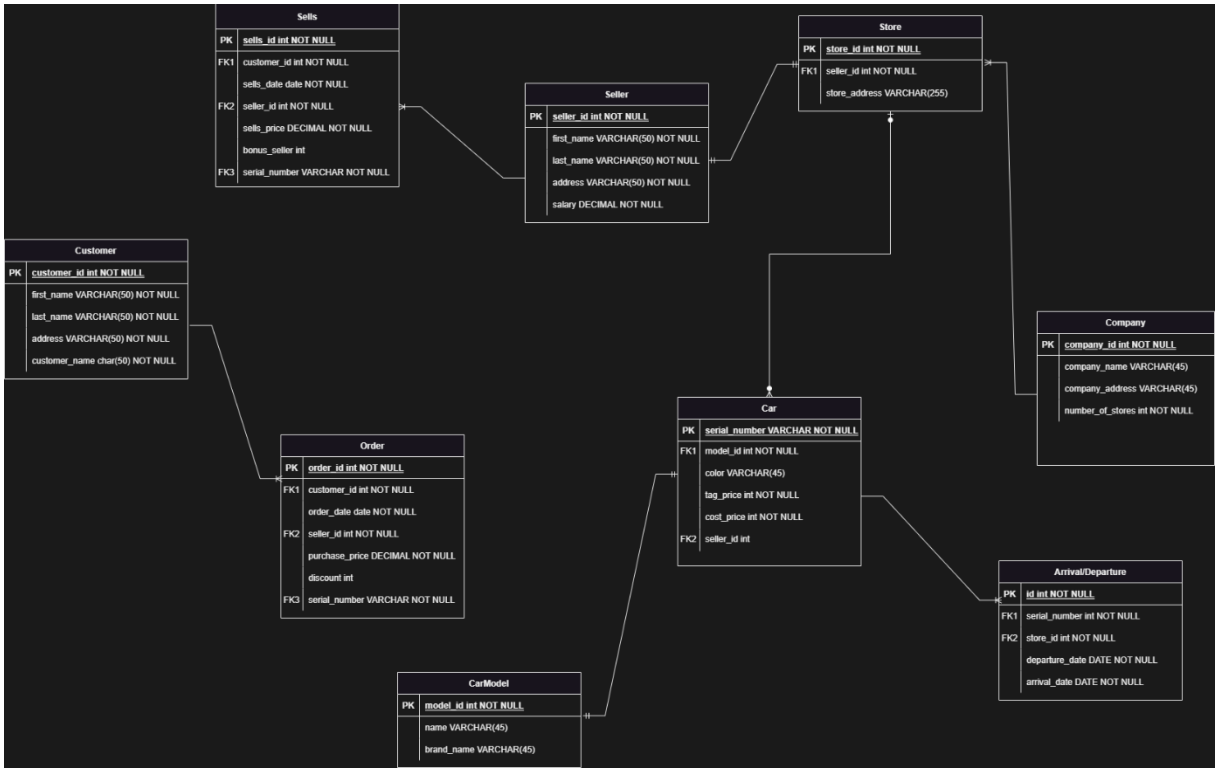
x) For each Cena's film (title and year), give the total actors' salaries.

```
mysql> SELECT TITLE,YEAR,SUM(D.SALARY) FROM FILM F JOIN DISTRIBUTION D ON F.FILM_NUMBER = D.FILM_NUMBER JOIN PERSON P ON P.PERSON_NUMBER = D.ACTOR_N
UMBER WHERE P.LASTNAME='Cena' GROUP BY F.TITLE,F.YEAR;
+-----------------------------------------------+------+---------------+
| TITLE                                         | YEAR | SUM(D.SALARY) |
+-----------------------------------------------+------+---------------+
| Spirited Away                                 | 2001 |         15000 |
| Attack on Titan: Crimson Bow and Arrow        | 2014 |         16500 |
| Neon Genesis Evangelion: The End of Evangelion | 1997 |         10000 |
| Your Name                                     | 2016 |          8000 |
+-----------------------------------------------+------+---------------+
4 rows in set (0.00 sec)

mysql>
```

Exercice 2

I) i) Give the conceptual data model of the above (E-R Diagram)



II) Convert this conceptual data model to **logical data model**. Hence, write the following SQL queries

- COMPANY (CompanyId (PK, INT), CompanyName (VARCHAR), SellerID (FK, VARCHAR), CompanyAddress (VARCHAR),NumberOfStore (INT, NULL))

- CAR (SerialNumber (PK, VARCHAR), Model (VARCHAR), Color (VARCHAR), TagPrice (DECIMAL), CostPrice (DECIMAL), SoldBy (FK, VARCHAR), StoredIn (FK, VARCHAR))

- CUSTOMER (CustomerID (PK, VARCHAR), LastName (VARCHAR), FirstName (VARCHAR), Address (VARCHAR))

- SELLER (SellerID(PK, VARCHAR), LastName(VARCHAR)

, FirstName (VARCHAR), Address (VARCHAR), Salary (DECIMAL))

- STORE (StoreID (PK, INT), StoreAddress (VARCHAR), SellerID (FK, VARCHAR))
- CAR_MODEL (ModelId (PK, INT), Name (VARCHAR), BrandName (VARCHAR))
- SELLS (SerialNumber (FK, VARCHAR), SellerID (FK, VARCHAR), SellsPrice (DECIMAL), SaleDate (DATE))
- CUSTOMER_ORDER (CustomerID (FK, VARCHAR), SerialNumber (FK, VARCHAR), PurchaseDate (DATE))
- ARRIVAL_OR_DEPARTURE_INFO (Id (PK, INT) , SerialNumber (FK, VARCHAR), StoreID (FK, VARCHAR), ArrivalDate (DATE),DepartureDate (DATE))

iii) Give the list of cars and serial numbers sold after the 1st of January 2023.

```
SELECT SerialNumber, ModelID
FROM Sale
JOIN Car ON Sale.SerialNumber = Car.SerialNumber
WHERE SaleDate > '2023-01-01';
```

iv) Give the name of the car with the highest sales

```
SELECT Brand, Name
FROM CarModel
WHERE ModelID = ( SELECT ModelID FROM Sale
    GROUP BY ModelID
    ORDER BY SUM(PurchasePrice - CostPrice) DESC
    LIMIT 1
);
```

v) Give the seller with the lowest discount

```
SELECT FirstName, LastName
FROM Seller
WHERE SellerID = ( SELECT SellerID FROM Sale
```

```
    ORDER BY Discount ASC
    LIMIT 1
);
```

vi) Give the profit of each store for the month of December 2022

```
SELECT StoreID, SUM(PurchasePrice - CostPrice) AS Profit
FROM Sale JOIN Car ON Sale.SerialNumber = Car.SerialNumber
WHERE SaleDate >= '2022-12-01' AND SaleDate < '2023-01-01'
GROUP BY StoreID;
```

vii) Give the best customer (The customer who has paid in the most money to the company)

```
SELECT FirstName, LastName
FROM Customer
WHERE CustomerID = ( SELECT CustomerID FROM Sale GROUP BY CustomerID
    ORDER BY SUM(PurchasePrice) DESC
    LIMIT 1
);
```

viii) The brand of the car with the highest discount

```
SELECT Brand
FROM CarModel
WHERE ModelID = ( SELECT ModelID FROM Sale
    ORDER BY Discount DESC
    LIMIT 1
);
```

ix) The car with the highest tag price

```
SELECT Brand, Name
FROM CarModel
WHERE ModelID = ( SELECT ModelID FROM Car
    ORDER BY TagPrice DESC
    LIMIT 1
);
```