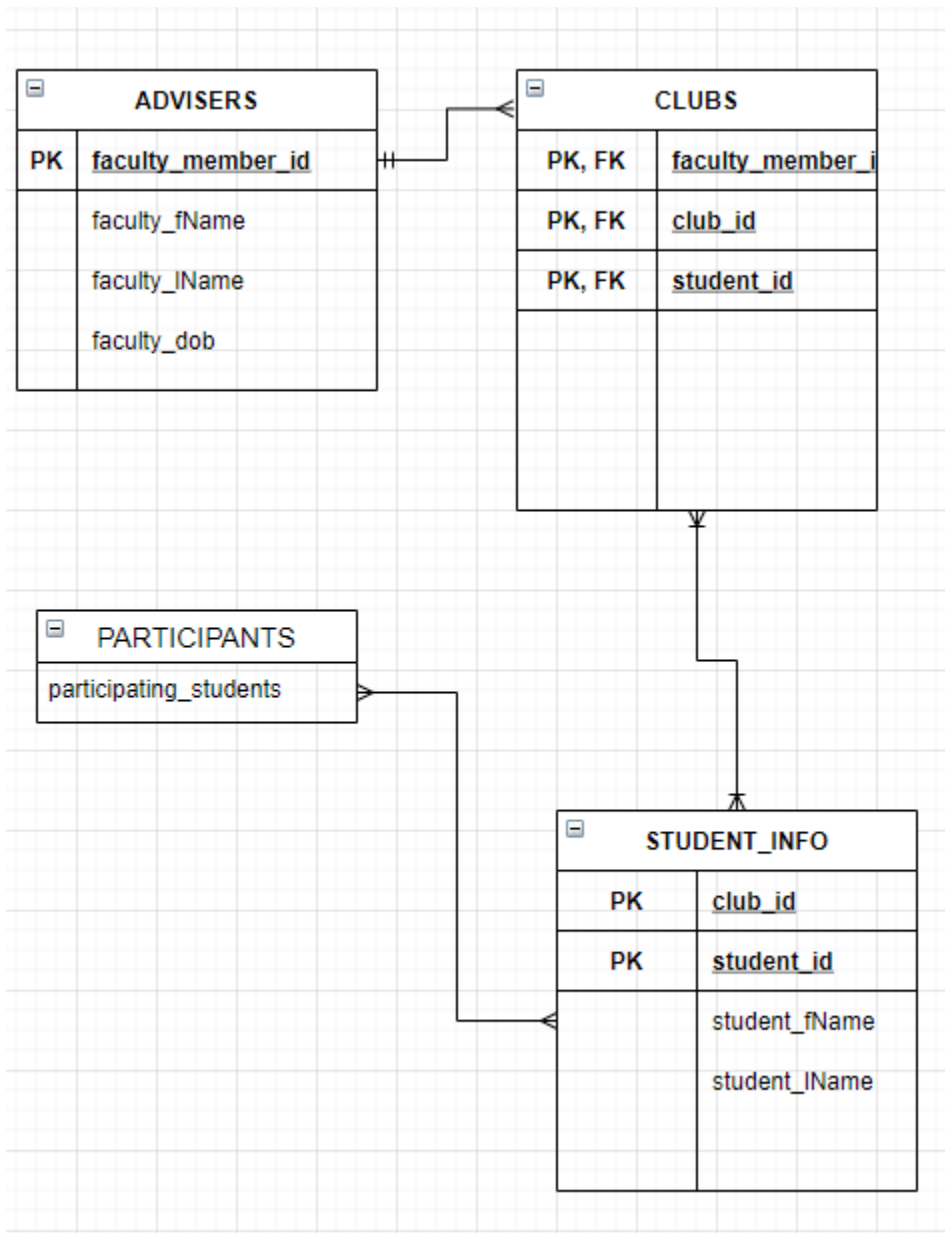


Lab 5

Part 1: ERDing My Heart Out



1)

2) TBA

3) ADVISERS: faculty_member_id
CLUBS: faculty_member_id, club_id, student_id
STUDENT_INFO: club_id, student_id

4) TBA

Part 2: From a View to a Query

```

1 • USE imdb;
2
3 • SELECT query_runner(now());
4
5 • DROP VIEW IF EXISTS avg_by_performer;
6 • CREATE VIEW avg_by_performer AS
7     SELECT
8         people.name AS 'Name',
9         AVG(ratings.rating) AS AverageRating,
10        query_runner() AS 'Query Runner / Date'
11    FROM people
12    JOIN crew ON people.person_id = crew.person_id
13    JOIN titles ON crew.title_id = titles.title_id
14    JOIN ratings ON titles.title_id = ratings.title_id
15    WHERE crew.category = 'actor' OR crew.category = 'actress'
16    GROUP BY people.name
17    ORDER BY AverageRating desc;
18
19 • SELECT * FROM avg_by_performer;

```

Filter Rows:
 Export:
 Wrap Cell Content:
 Fetch rows:

	Name	AverageRating	Query Runner / Date
▶	Yoshimasa Hosoya	9.899999618530273	Queried by Jericho Perdon on 2021-10-04 00:0...
	Kenshō Ono	9.899999618530273	Queried by Jericho Perdon on 2021-10-04 00:0...
	Tatsuhisa Suzuki	9.899999618530273	Queried by Jericho Perdon on 2021-10-04 00:0...
	Marina Inoue	9.899999618530273	Queried by Jericho Perdon on 2021-10-04 00:0...
	Yûto Uemura	9.899999618530273	Queried by Jericho Perdon on 2021-10-04 00:0...
	Matthew Needham	9.699999809265137	Queried by Jericho Perdon on 2021-10-04 00:0...
	Rob Wiethoff	9.699999809265137	Queried by Jericho Perdon on 2021-10-04 00:0...
	Jo Wyatt	9.699999809265137	Queried by Jericho Perdon on 2021-10-04 00:0...
	Jaimi Barbakoff	9.699999809265137	Queried by Jericho Perdon on 2021-10-04 00:0...
	Robert Sean Leonard	9.650000095367432	Queried by Jericho Perdon on 2021-10-04 00:0...
	Jami Reid-Quarrell	9.600000381469727	Queried by Jericho Perdon on 2021-10-04 00:0...
	Philip Barantini	9.600000381469727	Queried by Jericho Perdon on 2021-10-04 00:0...
	Radiohead	9.600000381469727	Queried by Jericho Perdon on 2021-10-04 00:0...
	Jeremy Dooley	9.600000381469727	Queried by Jericho Perdon on 2021-10-04 00:0...
	Clive Standen	9.600000381469727	Queried by Jericho Perdon on 2021-10-04 00:0...
	Baltasar Breki Samper	9.600000381469727	Queried by Jericho Perdon on 2021-10-04 00:0...

1) performer 1 x

Limit to 1000 rows

```

1  USE imdb;
2
3  • select query_runner(now());
4
5  • DROP VIEW IF EXISTS tv_seasons;
6  • CREATE VIEW tv_seasons AS
7  SELECT
8  titles.primary_title AS 'Show Title',
9  max(episodes.season_number) AS Seasons,
10 count(*) AS Episodes,
11 query_runner() AS 'Query Runner / Date'
12 FROM titles
13 JOIN episodes ON titles.title_id = episodes.show_title_id
14 GROUP BY show_title_id
15 ORDER BY Seasons ASC, Episodes DESC;
16
17 • SELECT * FROM tv_seasons;

```

<

Result Grid
Filter Rows:
Export:
Wrap Cell Content:

	Show Title	Seasons	Episodes	Query Runner / Date
▶	Late Night with Jimmy Fallon	0	379	Queried by Jericho Perdon on 2021-10-04 00:0...
	Days of Our Lives	1	4180	Queried by Jericho Perdon on 2021-10-04 00:0...
	EastEnders	1	2883	Queried by Jericho Perdon on 2021-10-04 00:0...
	Home and Away	1	2721	Queried by Jericho Perdon on 2021-10-04 00:0...
	Wan pîsu	1	357	Queried by Jericho Perdon on 2021-10-04 00:0...
	Teletubbies	1	142	Queried by Jericho Perdon on 2021-10-04 00:0...
	Dragon Ball	1	59	Queried by Jericho Perdon on 2021-10-04 00:0...
	Yu Yu Hakusho: Ghost Files	1	42	Queried by Jericho Perdon on 2021-10-04 00:0...
	Mahabharat	1	36	Queried by Jericho Perdon on 2021-10-04 00:0...
	Monster	1	28	Queried by Jericho Perdon on 2021-10-04 00:0...
	Samurai Champloo	1	12	Queried by Jericho Perdon on 2021-10-04 00:0...
	No Ordinary Family	1	11	Queried by Jericho Perdon on 2021-10-04 00:0...
	Future Diary	1	11	Queried by Jericho Perdon on 2021-10-04 00:0...
	John Doe	1	11	Queried by Jericho Perdon on 2021-10-04 00:0...
	The Adventures of Brisco C...	1	10	Queried by Jericho Perdon on 2021-10-04 00:0...
	Accidentally on Purpose	1	10	Queried by Jericho Perdon on 2021-10-04 00:0...

2) tv seasons 7

3) Tba

4) Tba

```
5) SELECT p1.name, category, primary_title, rating from people p1
JOIN crew USING(person_id)
JOIN titles USING(title_id)
JOIN ratings USING(title_id)
WHERE category = 'director'
AND title_id > (SELECT AVG(rating) from people p2
                JOIN CREW USING(person_id)
                JOIN ratings USING(title_id)
                WHERE p1.person_id.name = p2.person_id)

AND title_id IN (SELECT person_id from people
                JOIN crew USING(person_id)
                JOIN titles USING(title_id)
                GROUP BY person_id
                HAVING count(titles_id) > 2

ORDER BY rating DESC;
```

Part 3: Standard Operating Procedures (and Functions!)

Section 1: Procedural Thinking

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The script editor contains the following SQL code:

```
1 • USE imdb;
2
3 DELIMITER //
4 • CREATE TRIGGER ins_movie
5 BEFORE
6 INSERT ON titles
7 FOR EACH ROW
8 BEGIN
9     SET NEW.ended = 0;
10    SET NEW.premiered = 2021;
11 END //
12 DELIMITER ;
```

Below the script editor is an 'Output' section with a tab labeled 'Action Output'. It displays a table of execution results:

#	Time	Action	Message
✓ 1	18:21:31	USE imdb	0 row(s) affected
✓ 2	18:21:31	CREATE TRIGGER ins_movie BEFORE INSERT ON ti...	0 row(s) affected

1)

2) TBA

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and settings. The main editor displays a SQL script to create a trigger named `round_rating` that fires before insert on the `ratings` table. The script sets `NEW.votes` to 4975 if it is less than 4985, otherwise to 5000. The output pane at the bottom shows two successful actions: `USE imdb` and `CREATE TRIGGER round_rating BEFORE INSERT O...`, both with 0 rows affected.

```
1  USE imdb;
2
3  DELIMITER //
4  CREATE TRIGGER round_rating
5  BEFORE
6  INSERT ON ratings
7  FOR EACH ROW
8  BEGIN
9      IF NEW.votes < 4985 THEN
10         SET NEW.votes = 4975;
11     ELSE
12         SET NEW.votes = 5000;
13     END IF;
14 END //
15 DELIMITER ;
```

Output

Action Output

#	Time	Action	Message
✓ 1	18:26:29	USE imdb	0 row(s) affected
✓ 2	18:26:29	CREATE TRIGGER round_rating BEFORE INSERT O...	0 row(s) affected

3)

4) TBA

The screenshot shows a database IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The main editor displays the following SQL code:

```
1 • USE imdb;
2
3 • CREATE TABLE duplicated_people
4 (
5     person_id varchar(10),
6     name varchar(105),
7     born int(11),
8     died int(11),
9     PRIMARY KEY (person_id)
10 );
11
```

Below the editor is the 'Output' panel, which is set to 'Action Output'. It contains a table with the following data:

#	Time	Action	Message
✓ 1	18:29:55	USE imdb	0 row(s) affected
✓ 2	18:29:55	CREATE TABLE duplicated_people (person_id v...	0 row(s) affected

5)

Navigator

SCHEMAS

Filter objects

CI420

CIA

CIC

CIDolphinPods

CIMajors

imdb

Tables

akas

crew

duplicate_people

episodes

people

ratings

titles

Views

Stored Procedures

Functions

f() query_runner

LAB1PT3DB

LAB3PT3DB

PROJECTDB

saleco

Lab5_Almost_Titles

Lab5_Almost_Ratings

la

1

•

USE imdb;

2

3

•

ALTER TABLE people

4

ADD duplicate_insert_count int;

Output

Action Output

#

Time

Action

✓

1

18:33:08

USE imdb

✓

2

18:33:08

ALTER TABLE people ADD duplicate_

Administration

Schemas

Information

Table: people

Columns:

person_id

varchar

PK

name

varchar

born

int(11)

died

int(11)

duplicate_insert_count

int(11)

6)

Limit to 1000 rows

```
1 • USE imdb;
2
3 • UPDATE people
4   SET duplicate_insert_count = 0;
5 • SET sql_safe_updates = 0;
```

Output

Action Output

	#	Time	Action	Message
✓	1	18:35:53	USE imdb	0 row(s) affected
✓	2	18:35:53	UPDATE people SET duplicate_insert_count = 0	0 row(s) affected Rows
✓	3	18:35:53	SET sql_safe_updates = 0	0 row(s) affected

7)

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and settings. A dropdown menu is set to "Limit to 1000 rows". The main editor contains a SQL script with 14 lines of code. A blue dot is on line 4, and a cursor is on line 8. The script creates a trigger named "people_upd" that updates the "duplicate_insert_count" in the "people" table whenever a new row is inserted into the "duplicated_people" table.

```
1  USE imdb;
2
3  DELIMITER //
4  CREATE TRIGGER people_upd
5  BEFORE
6  INSERT on duplicated_people
7  FOR EACH ROW
8  BEGIN
9      UPDATE people
10     SET people.duplicate_insert_count = people.duplicate_insert_count + 1
11     WHERE people.person_id = NEW.person_id;
12
13 END //
14 DELIMITER ;
```

Below the editor is an "Output" pane. It has a tab labeled "Action Output". The output table has four columns: "#", "Time", "Action", and "Message". It contains two rows of execution results, both marked with green checkmarks.

#	Time	Action	Message
1	18:38:53	USE imdb	0 row(s) affected
2	18:38:53	CREATE TRIGGER people_upd BEFORE INSERT...	0 row(s) affected

8)

9) TBA