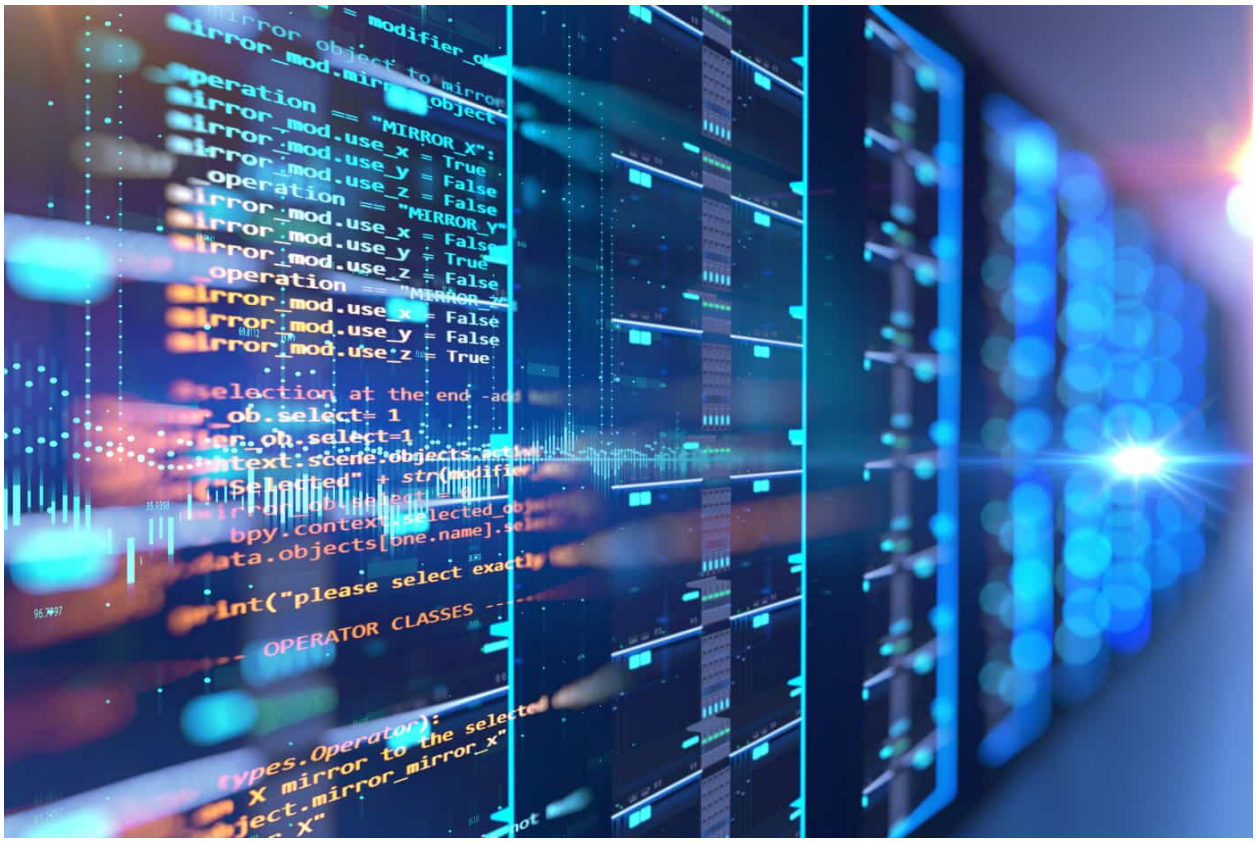
College Dublin
Computing • IT • Business

# Data Manipulation and Validation

*Integrated CA2, Databases, Aldana Louzan*



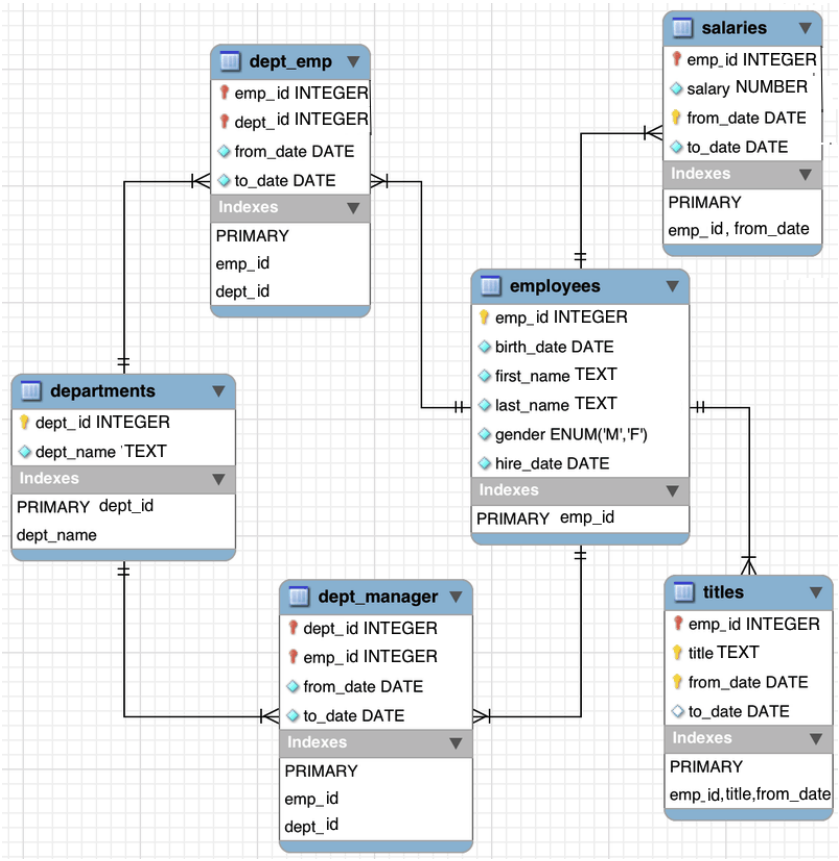

**Tolga Baris PINAR**

01.12.2022

Higher Diploma in Science in Computing

# INTRODUCTION

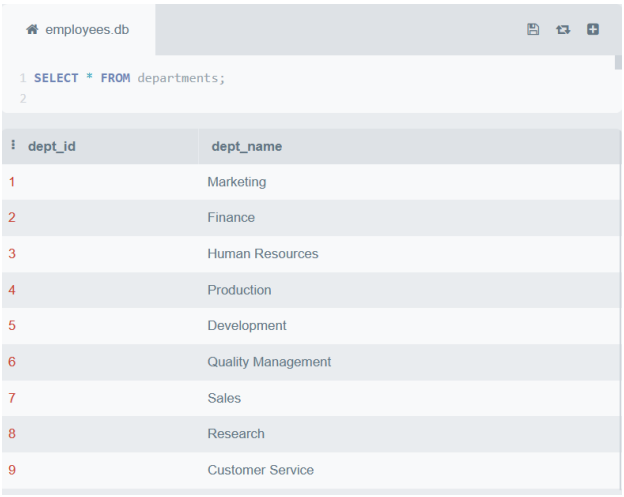
In this assignment, the employees sample database (created by Fusheng Wang and Carlo Zaniolo at Siemens Corporate Research), a large base of data spread over six separate tables and consisting of 4 million records in total that was created for system testing purposes. The following diagram provides an overview of the structure of the employees.db:



## 1.1 Databases CA Part 1

### 1.1.1 List all attributes present in the **departments** TABLE.

**CODE:** SELECT * FROM departments;



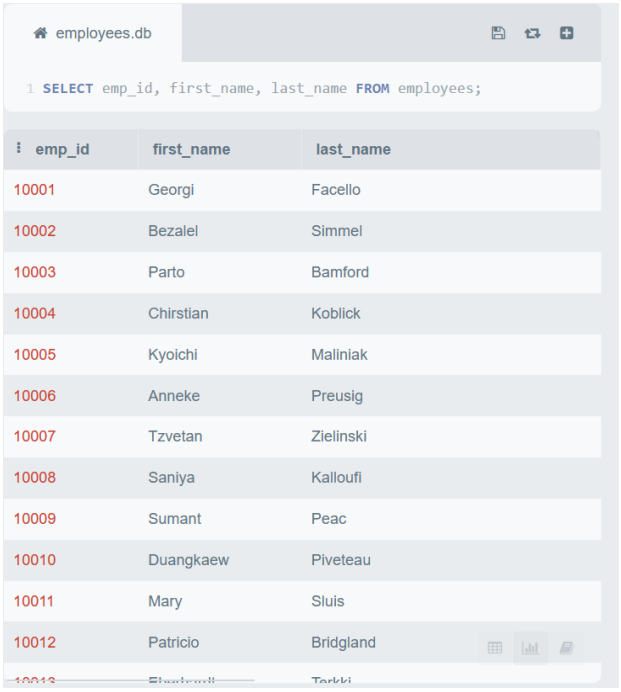
employees.db

```
1 SELECT * FROM departments;
2
```

| dept_id | dept_name |
|---|---|
| 1 | Marketing |
| 2 | Finance |
| 3 | Human Resources |
| 4 | Production |
| 5 | Development |
| 6 | Quality Management |
| 7 | Sales |
| 8 | Research |
| 9 | Customer Service |

### 1.1.2. List all **employee IDs** of all past/current employees, their **first** and **last names**.

**CODE:** SELECT emp_id, first_name, last_name FROM employees;



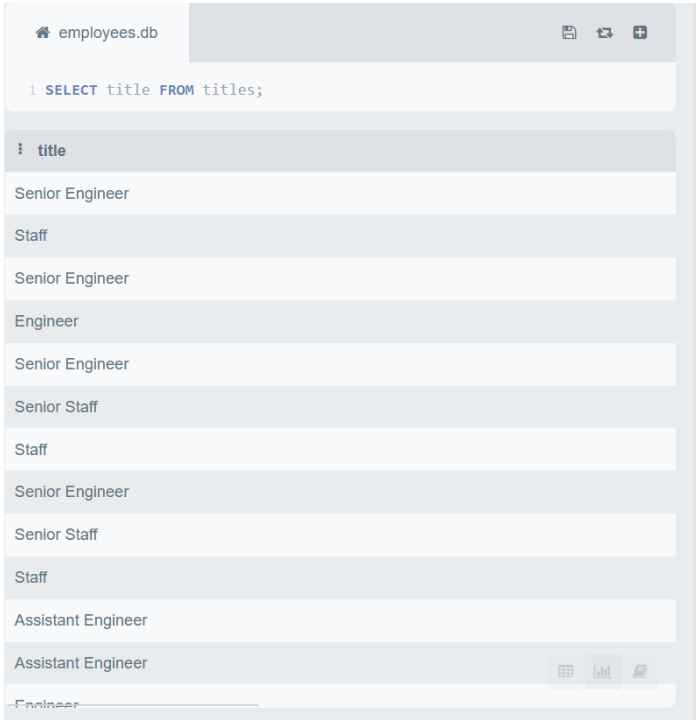
employees.db

```
1 SELECT emp_id, first_name, last_name FROM employees;
```

| emp_id | first_name | last_name |
|---|---|---|
| 10001 | Georgi | Facello |
| 10002 | Bezalel | Simmel |
| 10003 | Parto | Bamford |
| 10004 | Chirstian | Koblick |
| 10005 | Kyoichi | Maliniak |
| 10006 | Anneke | Preusig |
| 10007 | Tzvetan | Zielinski |
| 10008 | Saniya | Kalloufi |
| 10009 | Sumant | Peac |
| 10010 | Duangkaew | Piveteau |
| 10011 | Mary | Sluis |
| 10012 | Patricio | Bridgland |

### 1.1.3. List all department **titles** present in the database.

**CODE:** SELECT title FROM titles;

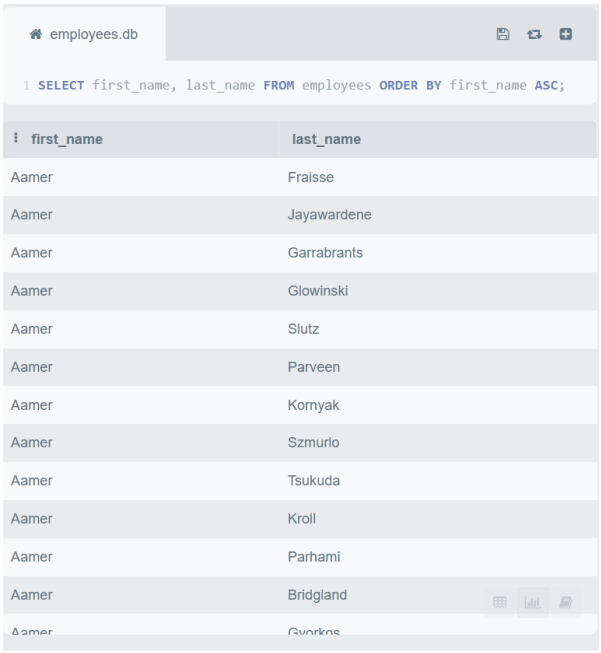



### 1.1.4. List all **unique job titles** found in the database, and **order them alphabetically**.

**CODE:** SELECT DISTINCT title FROM titles ORDER BY title ASC;

### 1.1.5. List all past/current **employees' names ordered alphabetically** in ascending order, i.e. first name and last name in alphabetical order.

**CODE:** SELECT first_name, last_name FROM employees ORDER BY first_name ASC;

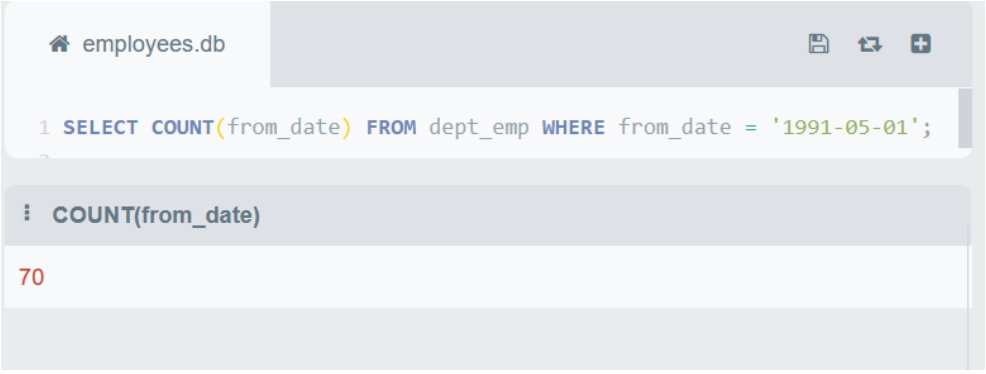



| first_name | last_name |
|---|---|
| Aamer | Fraisse |
| Aamer | Jayawardene |
| Aamer | Garrabrants |
| Aamer | Glowinski |
| Aamer | Slutz |
| Aamer | Parveen |
| Aamer | Kornyak |
| Aamer | Szmurlo |
| Aamer | Tsukuda |
| Aamer | Kroll |
| Aamer | Parhami |
| Aamer | Bridgland |
| Aamer | Gvorkos |

## 1.2 Database CA Part 2

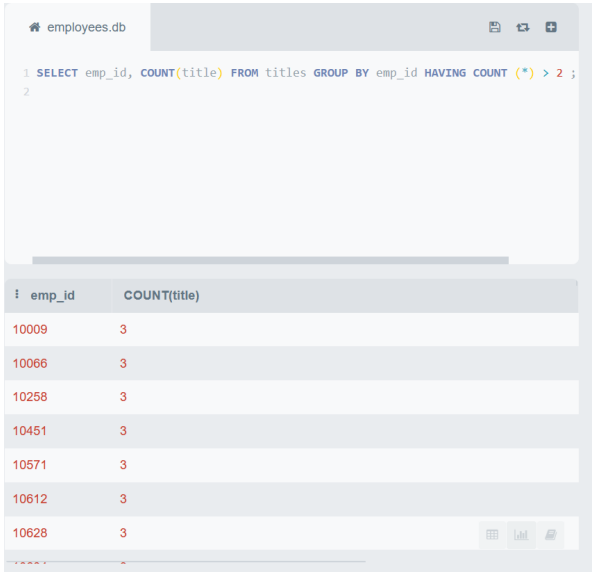
### 1.2.1 The number of all employees that started on **1991-05-01**.

**CODE:** SELECT COUNT(from_date) FROM dept_emp WHERE from_date = '1991-05-01';



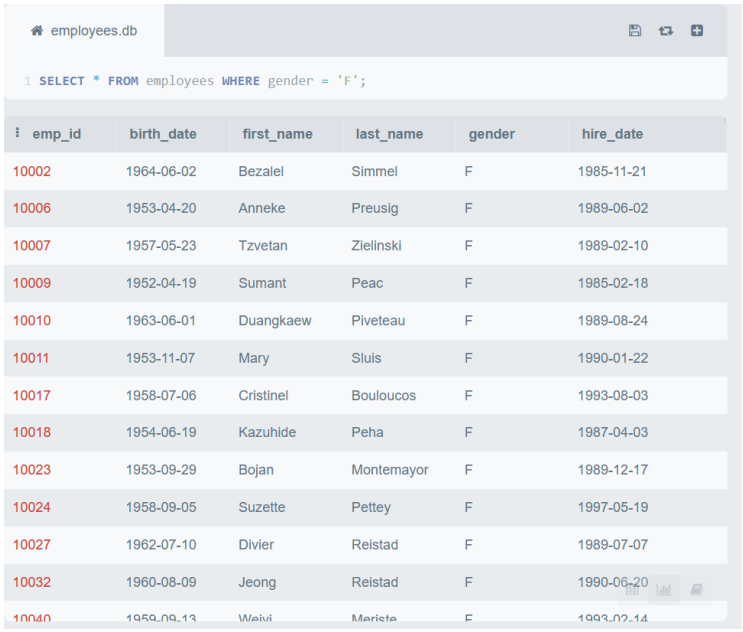
### 1.2.2 List all **emp_no** who have had strictly **more than 2 titles** and display **the total number of the titles** they have had.

**CODE:** SELECT emp_id, COUNT(title) FROM titles GROUP BY emp_id HAVING COUNT (*) > 2 ;



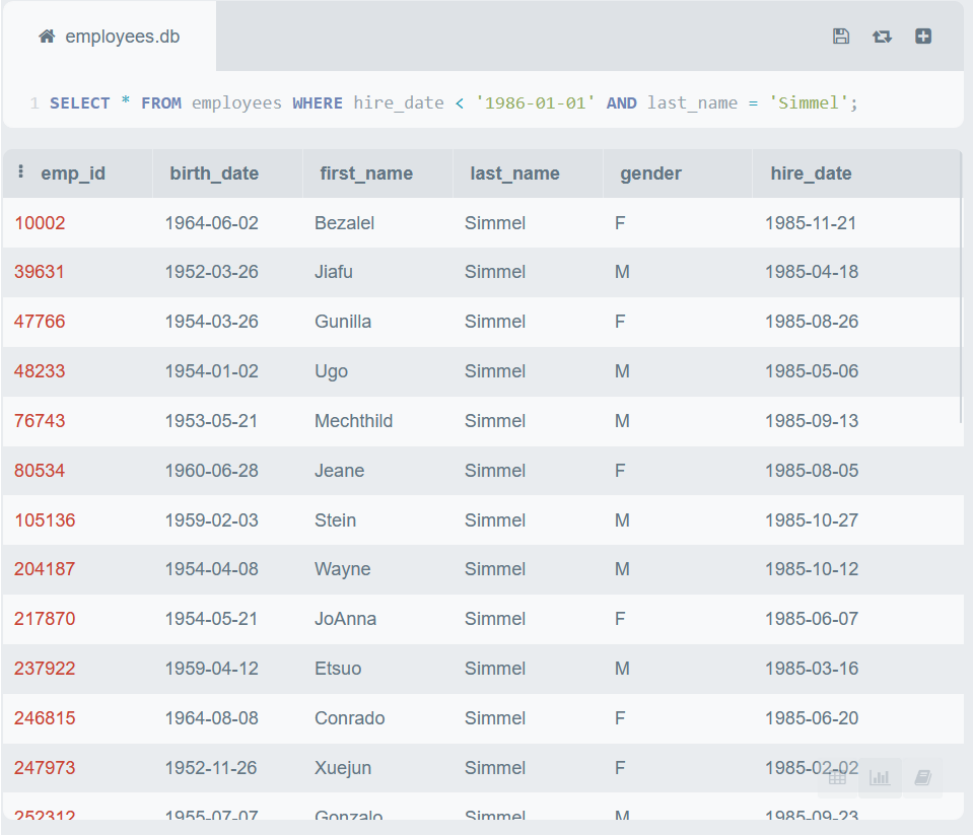

### 1.2.3 List **female employees** (past/current) together with all other relation attributes.

**CODE:** SELECT * FROM employees WHERE gender = 'F';



### 1.2.4 List past/current **employees** hired prior to **1986-01-01** with the surname **Simmel**.

**CODE:** SELECT * FROM employees WHERE hire_date < '1986-01-01' AND last_name = 'Simmel';

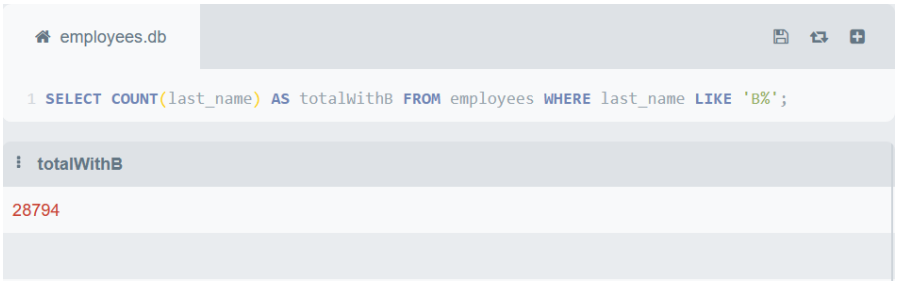



| emp_id | birth_date | first_name | last_name | gender | hire_date |
|---|---|---|---|---|---|
| 10002 | 1964-06-02 | Bezalel | Simmel | F | 1985-11-21 |
| 39631 | 1952-03-26 | Jiafu | Simmel | M | 1985-04-18 |
| 47766 | 1954-03-26 | Gunilla | Simmel | F | 1985-08-26 |
| 48233 | 1954-01-02 | Ugo | Simmel | M | 1985-05-06 |
| 76743 | 1953-05-21 | Mechthild | Simmel | M | 1985-09-13 |
| 80534 | 1960-06-28 | Jeane | Simmel | F | 1985-08-05 |
| 105136 | 1959-02-03 | Stein | Simmel | M | 1985-10-27 |
| 204187 | 1954-04-08 | Wayne | Simmel | M | 1985-10-12 |
| 217870 | 1954-05-21 | JoAnna | Simmel | F | 1985-06-07 |
| 237922 | 1959-04-12 | Etsuo | Simmel | M | 1985-03-16 |
| 246815 | 1964-08-08 | Conrado | Simmel | F | 1985-06-20 |
| 247973 | 1952-11-26 | Xuejun | Simmel | F | 1985-02-02 |
| 252312 | 1955-07-07 | Gonzalo | Simmel | M | 1985-09-23 |

### 1.2.5 How many past/current **employees' last names** begin with the capital letter B?

Use a column alias **total with B** to output your results.

**CODE:** SELECT COUNT(last_name) AS totalWithB FROM employees WHERE last_name LIKE 'B%';




| totalWithB |
|---|
| 28794 |

### 1.2.6 Create a new table called **emp_training** with 3 columns:

• **trainer_no:** this should be the primary key and is of type integer and is an auto-increment.

• **first_name:** this data type is **varchar(30)** and should not be NULL

• **last_name:** this data type is **varchar(30)** and should not be NULL

• **t_module:** this data type is **varchar(20)**

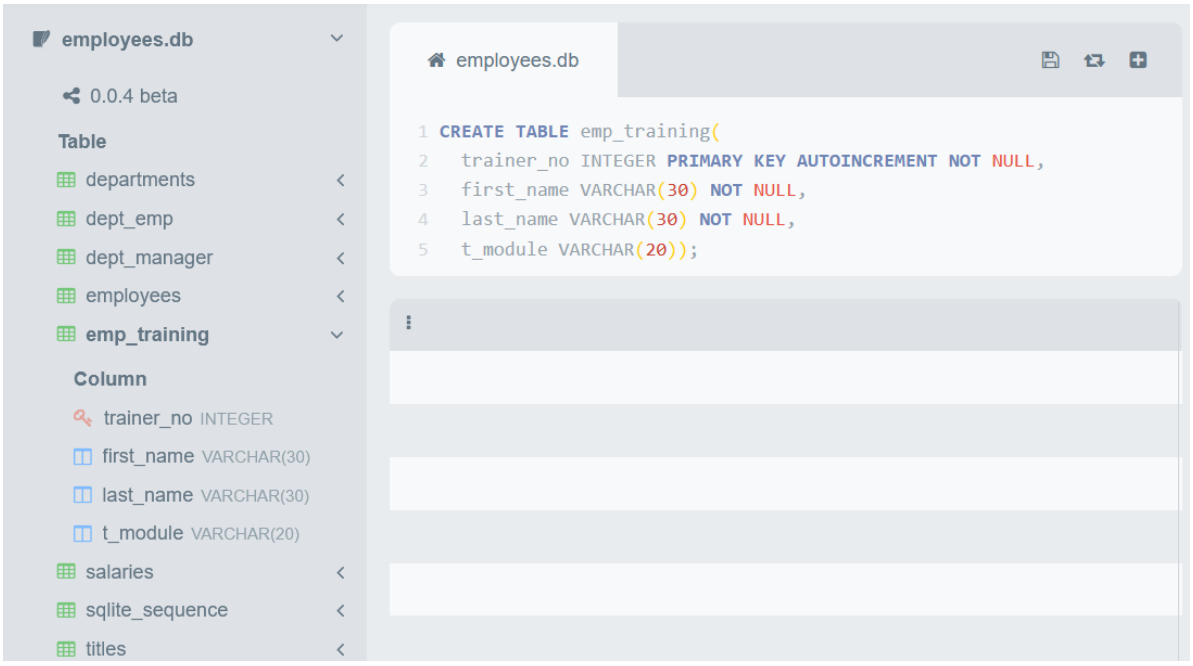
**CODE:** CREATE TABLE emp_training (

trainer_no INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

first_name VARCHAR(30) NOT NULL,

last_name VARCHAR(30) NOT NULL,

t_module VARCHAR(20) );



### 1.2.7 Insert 2 new rows into the **emp_training** table:

**Row 1:** fname: **Joe**

lname: **Bloggs**

module: **Google Docs**

**Row 2:** fname: **Fred**

lname: **Bloggs**

module: **Google Sheets**

**CODE:**

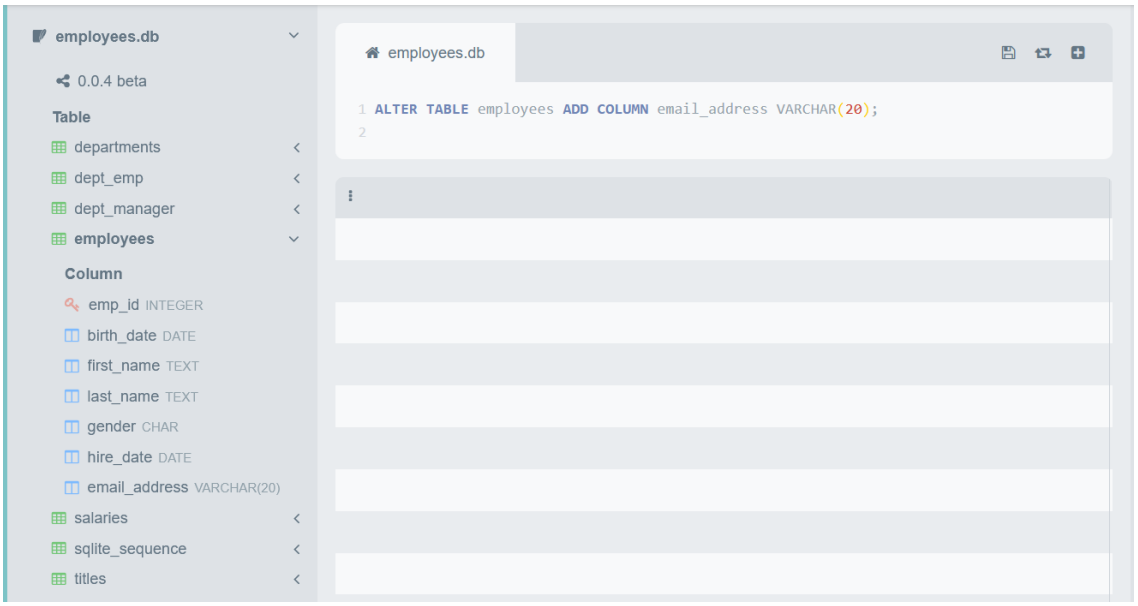
```
INSERT INTO emp_training (first_name, last_name, t_module) VALUES ('Joe', 'Bloggs', 'Google Docs');

INSERT INTO emp_training (first_name, last_name, t_module) VALUES ('Fred', 'Bloggs', 'Google Sheets');
```


### 1.2.8 The organisation no longer wishes to record the employees training within the database. Therefore, delete the newly created **emp_training** table.

**CODE:** DROP TABLE IF EXISTS emp_training;

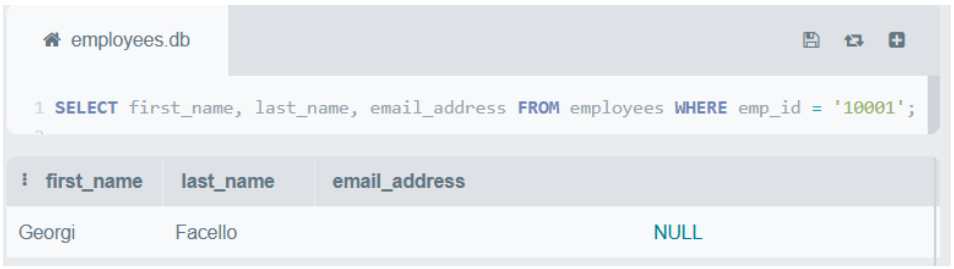
1.2.9 Alter the **employees** table to include an **email_address** field of type **varchar(20)**.

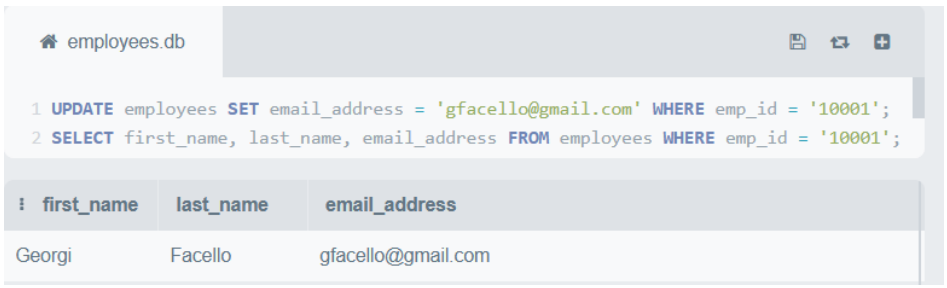
**CODE:** ALTER TABLE employees ADD COLUMN email_address VARCHAR(20);




1.2.10 Update the email address of **Georgi Facello** to **gfacello@gmail.com**, where **emp_no** equals to **10001**.

**CODE:** UPDATE employees SET email_address = 'gfacello@gmail.com' WHERE emp_id = '10001';



## 1.3 Database CA Part 3

### 1.3.1 List the number of male managers and female managers who work for each department. Make sure to display the gender, the number of employees (renamed as num_empGender) and dept_no, ordered by department number in an ascendant order.

**CODE:**

```
SELECT dept_id,
    sum( gender = 'F' ) AS num_empFemale,
    sum( gender = 'M' ) AS num_empMale
FROM dept_manager dm INNER JOIN employees e ON dm.emp_id = e.emp_id
GROUP BY dept_id ORDER BY dept_id ASC;
```

employees.db

```
1 SELECT dept_id,
2     sum( gender = 'F' ) AS num_empFemale,
3     sum( gender = 'M' ) AS num_empMale
4 FROM dept_manager dm INNER JOIN employees e ON dm.emp_id = e.emp_id
5 GROUP BY dept_id ORDER BY dept_id ASC;
```

| dept_id | num_empFemale | num_empMale |
|---|---|---|
| 1 | 0 | 2 |
| 2 | 1 | 1 |
| 3 | 2 | 0 |
| 4 | 2 | 2 |
| 5 | 1 | 1 |
| 6 | 3 | 1 |
| 7 | 0 | 2 |
| 8 | 1 | 1 |
| 9 | 3 | 1 |

1.3.2 List the average salary of male and female employees whose title is "Technique Leader". In your result table should appear, gender, average salary named as avg_salary and title.

**CODE:**

SELECT gender, AVG(salary) AS avg_salary, title FROM titles tile INNER JOIN employees e, salaries sal ON tile.emp_id = e.emp_id

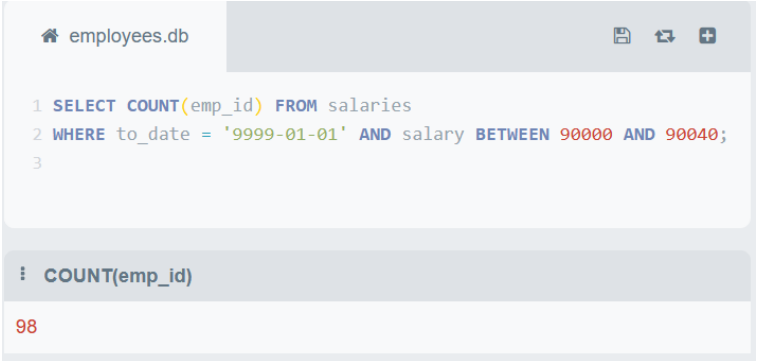
AND tile.emp_id = sal.emp_id AND e.emp_id = sal.emp_id

WHERE title = 'Technique Leader' GROUP BY gender;


1.3.3 The number of employees that have a current salary (i.e., **to_date** equals to **9999-01-01**) between **90000** and **90040**.

**CODE:** SELECT COUNT(emp_id) FROM salaries

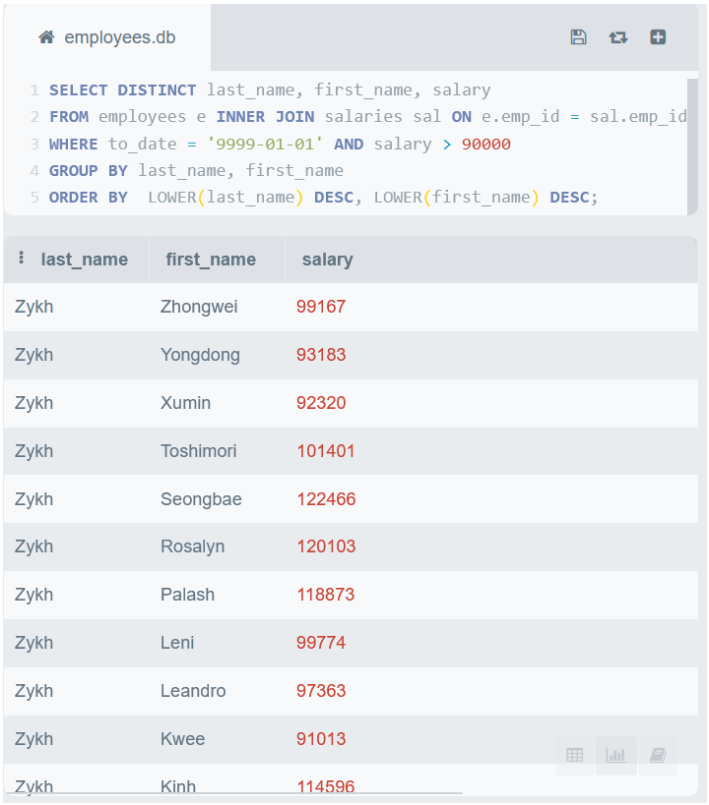
WHERE to_date = '9999-01-01' AND salary BETWEEN 90000 AND 90040;



1.3.4 List all unique employees' last and first names (using **GROUP BY** method) that have a current salary (i.e., **to_date** equals to **9999-01-01**) greater than **90000**, outputting both names in descending order (sort by the last name first and then the first name) and also displaying their current salaries (using the **INNER JOIN** method).

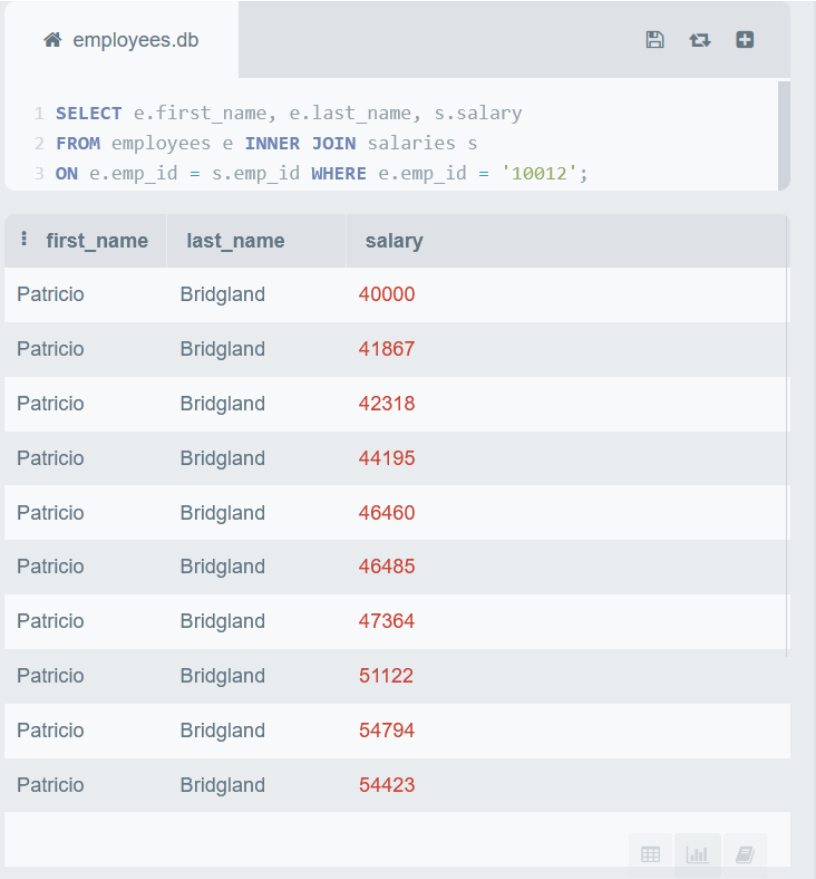
**CODE:**

```
SELECT DISTINCT last_name, first_name, salary

FROM employees e INNER JOIN salaries sal ON e.emp_id = sal.emp_id

WHERE to_date = '9999-01-01' AND salary > 90000

GROUP BY last_name, first_name

ORDER BY  LOWER(last_name) DESC, LOWER(first_name) DESC;

/* Lower should be used since there are people with the surname called " dAstous" which basically starts with a lowercase letter*/
```



### 1.3.5 First name, last name, all salary dates and related amounts for the employee with employee number **10012**.

**CODE:** SELECT e.first_name, e.last_name, s.salary FROM employees e INNER JOIN salaries s ON e.emp_id = s.emp_id WHERE e.emp_id = '10012';




| first_name | last_name | salary |
|---|---|---|
| Patricio | Bridgland | 40000 |
| Patricio | Bridgland | 41867 |
| Patricio | Bridgland | 42318 |
| Patricio | Bridgland | 44195 |
| Patricio | Bridgland | 46460 |
| Patricio | Bridgland | 46485 |
| Patricio | Bridgland | 47364 |
| Patricio | Bridgland | 51122 |
| Patricio | Bridgland | 54794 |
| Patricio | Bridgland | 54423 |

### 1.3.6 In relation to the table named salaries in **Figure 1** above. Answer in text:

a) What is the **degree** of this table?

- The table named salaries has four degrees, which are emp_id, salary, from_date and to_date, at total.

b) What column(s), if any, make(s) up the **primary key**?

-emp_id and from_date are the primary keys in this table.

c) What column(s), if any, make(s) up the **foreign key**?

-emp_id is the only foreign key in this table.

1.3.7 In the given schema, the tables dept_emp, dept_manager, salaries, titles have composite keys.

Explain for each relation why this is the case? Support your answer with appropriate references.

-TABLE **dept_emp**: This table has a composite key, consists of **emp_id** FOREIGN KEY and **dept_id** FOREIGN KEY . This composite key uses the combination of the PRIMARY KEY of the **employees** TABLE and **departments** TABLE, thus one can identify an employee, retrieve the data of the department an employee works in, and the date an employee starts and ends working in that department.

-TABLE **dept_manager**: This table has a composite key, consists of **emp_id** FOREIGN KEY (also a PRIMARY KEY in this table) and **dept_id** FOREIGN KEY . This composite key uses the combination of the PRIMARY KEY of the **employees** TABLE and **departments** TABLE, and therefore one can identify an employee, who is a manager, retrieve the data of the department a manager manages, and the managing period of a department.

-TABLE **salaries**: This table has a composite key, consists of **emp_id** FOREIGN KEY and **from_date** PRIMARY KEY. This is a composite key, but also a compound key since it consists of one primary and one foreign key. This composite key uses the combination of the PRIMARY KEY of the **employees** TABLE and its own PRIMARY KEY so that one can identify an employee, retrieve the data of the salary an employee gets, and the period an employee gets a salary. However

-TABLE **titles**: This table has a composite key, which is also a compound key, which consists of **emp_id** FOREIGN KEY (also a PRIMARY KEY in this table), **title** PRIMARY KEY and **from_date** PRIMARY KEY. This composite key uses the combination of the PRIMARY KEY of the **employees** TABLE and its own PRIMARY KEYs so that one can identify an employee, retrieve the data of a title of an employee, and the period an employee holds the title.

## SQL QUERIES SUMMARY:

1.1.1 SELECT * FROM departments;

1.1.2 SELECT emp_id, first_name, last_name FROM employees;

1.1.3 SELECT title FROM titles;

1.1.4 SELECT DISTINCT title FROM titles ORDER BY title ASC;

1.1.5 SELECT first_name, last_name FROM employees ORDER BY first_name ASC;

1.2.1 SELECT COUNT(from_date) FROM dept_emp WHERE from_date = '1991-05-01';

1.2.2 SELECT emp_id, COUNT(title) FROM titles GROUP BY emp_id HAVING COUNT (*) > 2 ;

1.2.3 SELECT * FROM employees WHERE gender = 'F';

1.2.4 SELECT * FROM employees WHERE hire_date < '1986-01-01' AND last_name = 'Simmel';

1.2.5 SELECT COUNT(last_name) AS totalWithB FROM employees WHERE last_name LIKE 'B%';

1.2.6 CREATE TABLE emp_training(trainer_no INTEGER

PRIMARY KEY AUTOINCREMENT NOT NULL, first_name

VARCHAR(30) NOT NULL, last_name VARCHAR(30) NOT

NULL, t_module VARCHAR(20));

1.2.7 INSERT INTO emp_training (first_name, last_name, t_module) VALUES ('Joe', 'Bloggs', 'Google Docs');

INSERT INTO emp_training (first_name, last_name, t_module) VALUES ('Fred', 'Bloggs', 'Google Sheets');

1.2.8 DROP TABLE IF EXISTS emp_training;

1.2.9 ALTER TABLE employees ADD COLUMN email_address VARCHAR(20);

1.2.10 UPDATE employees SET email_address = 'gfacello@gmail.com' WHERE emp_id = '10001';

1.3.1 SELECT dept_id, sum( gender = 'F' ) AS num_empFemale, sum( gender = 'M' ) AS num_empMale FROM dept_manager dm INNER JOIN employees e ON dm.emp_id = e.emp_id GROUP BY dept_id ORDER BY dept_id ASC;

1.3.2 SELECT gender, AVG(salary) AS avg_salary, title FROM titles tile INNER JOIN employees e, salaries sal ON tile.emp_id = e.emp_id AND tile.emp_id = sal.emp_id AND e.emp_id = sal.emp_id WHERE title = 'Technique Leader' GROUP BY gender;

1.3.3 SELECT COUNT(emp_id) FROM salaries WHERE to_date = '9999-01-01' AND salary BETWEEN 90000 AND 90040;

1.3.4 SELECT DISTINCT last_name, first_name, salary FROM employees e INNER JOIN salaries sal ON e.emp_id = sal.emp_id WHERE to_date = '9999-01-01' AND salary > 90000 GROUP BY last_name, first_name ORDER BY LOWER(last_name) DESC, LOWER(first_name) DESC;

1.3.5 SELECT e.first_name, e.last_name, s.salary FROM employees e INNER JOIN salaries s ON e.emp_id = s.emp_id WHERE e.emp_id = '10012';

## REFERENCES

1. www.w3schools.com. (n.d.). *MySQL Joins*. [online] Available at: https://www.w3schools.com/mysql/mysql_join.asp.
2. Otieno, J. (n.d.). *Ambiguous Column Name with SQL Join Query*. [online] Available at:

https://linuxhint.com/ambiguous-column-name-with-sql-join-query/ [Accessed 1 Dec. 2022].

3. sqlite.org. (n.d.). *Built-in Aggregate Functions*. [online] Available at: https://sqlite.org/lang_aggfunc.html [Accessed 1 Dec. 2022].
4. sqlite.org. (n.d.). *SQLite Documentation*. [online] Available at: https://sqlite.org/docs.html.
5. www.bestinterviewquestion.com. (n.d.). *What is the degree of a table in MySQL? - Best Interview Question*. [online] Available at: https://www.bestinterviewquestion.com/question/what-is-the-degree-of-a-table-in-mysql-5xe4y5359j4 [Accessed 1 Dec. 2022].
6. K, S. and asamy (n.d.). *What is degree of a relation in dbms*. [online] Available at: https://www.exploredatabase.com/2017/02/what-is-degree-of-relation-table-in-dbms.html.
7. SolveXia (2019). *5 Top Tips for Data Manipulation*. [online] www.solvexia.com. Available at: https://www.solvexia.com/blog/5-top-tips-for-data-manipulation.
8. Stack Overflow. (n.d.). *indexing - Differences between INDEX, PRIMARY, UNIQUE, FULLTEXT in MySQL?* [online] Available at: https://stackoverflow.com/questions/707874/differences-between-index-primary-unique-fulltext-in-mysql [Accessed 1 Dec. 2022].
9. Stack Overflow. (n.d.). *mysql - What is difference between primary index and secondary index exactly?* [online] Available at: https://stackoverflow.com/questions/20824686/what-is-difference-between-primary-index-and-secondary-index-exactly.
10. dev.mysql.com. (n.d.). *MySQL :: MySQL 8.0 Reference Manual :: MySQL Glossary*. [online] Available at: https://dev.mysql.com/doc/refman/8.0/en/glossary.html.

11. www.ibm.com. (2022). *Defining Composite Primary and Foreign Keys*. [online] Available at: https://www.ibm.com/docs/en/informix-servers/14.10?topic=format-defining-composite-primary-foreign-keys.
12. www.youtube.com. (n.d.). *SQL Index |¦| Indexes in SQL |¦| Database Index*. [online] Available at: https://www.youtube.com/watch?v=fsG1XaZEa78.
13. www.youtube.com. (n.d.). *SQL Joins Explained |¦| Joins in SQL |¦| SQL Tutorial*. [online] Available at: https://www.youtube.com/watch?v=9yeOJ0ZMUYw.
14. Stack Overflow. (n.d.). *sql - What do the mysql workbench column icons mean*. [online] Available at: https://stackoverflow.com/questions/10778561/what-do-the-mysql-workbench-column-icons-mean.
15. Studytonight (2019). *Concept of Keys in DBMS - Super, Primary, Candidate, Foreign Key, etc*. *YouTube*. Available at: https://www.youtube.com/watch?v=p3yJZH8_bsc.
16. Aldana Louza(2022) described Relational Databases(Moodle)
17. Aldana Louza(2022) described SQL (Moodle)
18. Aldana Louza(2022) described Classic Models
19. Aldana Louza(2022) described Data Manipulation Language (Moodle)
20. Aldana Louza(2022) described Data Definition Language (Moodle)
21. Aldana Louza(2022) described SQL & Data Query Language (Moodle)