LAB 1 (3/10/2024)

Lab Description

This lab reviews your knowledge of database management systems and their important aspects. This lab involves knowledge of Entity relationship diagrams and using the correct cardinalities, which is important for the whole database design and integrity. A good understanding of the business needs can help us to build an Entity relationship diagram which is essential to represent the relationship between entities.

— LAB ANSWERS —

Q1. Which of the following steps occurs first in the process of building an ERD based on a scenario?

a. Develop the initial ERD.
b. Create a detailed narrative of the organisation's description of operations.
c. Identify the attributes and primary keys that adequately describe the entities.
d. Identify the business rules based on the description of operations.

Answer:

Identify the business rules based on the description of operations

Q2. A student can attend 5 modules. Different lecturers can offer the same module. The relationship of students → lecturers is a _____ relationship.

a. Many-to-many (M: M)
b. One-to-many (1:M)
c. One-to-one (1:1)
d. Many-to-one (M:1)
e. Many-to-zero OR one (M:0-1)
f. One-to-one OR many (1:1-M)
g. Many-to-zero

Answer:

Many-to-many (M:M)

Q3. What would be the cardinality symbol for the relationship identified in Q2?

a. 
b. 
c. 
d. 
e. 
f. 

Answer:

c . 

Q4. Which of the following statements best describes the function of an entity relation model?

a. An ER model is concerned primarily with the view of the attributes that will be used in physical implementation
b. An ER model is concerned primarily with a physical implementation of the data and secondly with the logical view
c. An ER model provides a view of the logic of the data and not the physical implementation
d. An ER model is entirely concerned with modelling the physical implementation

Answer :

An ER model provides a view of the logic of the data and not the physical implementation

Q5. Consider Figure 1 representing instances of a relationship between EMPLOYEE and the DEPARTMENT that the employees work in.

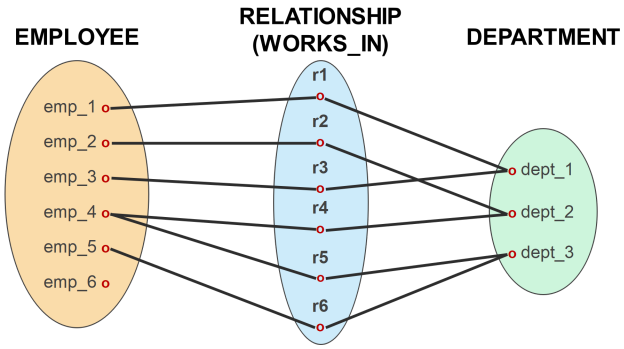


Figure.1

Which of the following relationships are solved and represented by the above figure?

a. **1:1** relationship between EMPLOYEE and DEPARTMENT and total participation from EMPLOYEE and total participation from DEPARTMENT
b. **1:M** relationship between EMPLOYEE and DEPARTMENT and partial participation from EMPLOYEE and partial participation from DEPARTMENT
c. **M:M** relationship between EMPLOYEE and DEPARTMENT and partial participation from EMPLOYEE and total participation from DEPARTMENT
d. **M:1** relationship between EMPLOYEE and DEPARTMENT and total participation from EMPLOYEE and total participation from DEPARTMENT

Answer:

M:M relationship between EMPLOYEE and DEPARTMENT and partial participation from EMPLOYEE and total participation from DEPARTMENT

Q6. Suppose we map the following ERD (Figure 2) to the relations T1(A, B) and T2(B, C). The CREATE table statement for T2 is defined as the following:

```
CREATE TABLE T2(B SERIAL PRIMARY KEY, C INT).
```

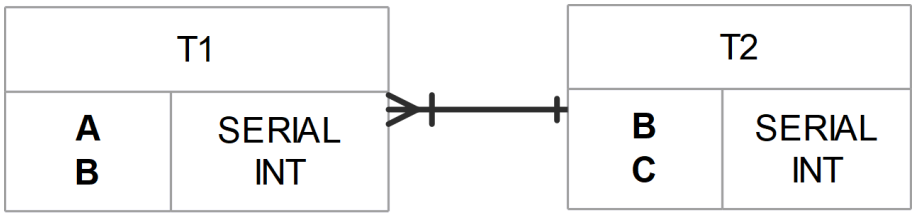


Figure. 2

Which one of the following create table statements would be correct for T1, enforcing the FK and relationship?

a. `CREATE TABLE T1(A SERIAL PRIMARY KEY UNIQUE, B INT, FOREIGN KEY NOT NULL);`
b. `CREATE TABLE T1(A SERIAL PRIMARY KEY, B INT NOT NULL REFERENCES T2(B));`
c. `CREATE TABLE T1(A SERIAL UNIQUE, B REFERENCES T2(B));`
d. `CREATE TABLE T1(A SERIAL, B INT);`

Answer:

```
b.CREATE TABLE T1(A SERIAL PRIMARY KEY, B INT NOT NULL REFERENCES T2(B));
```

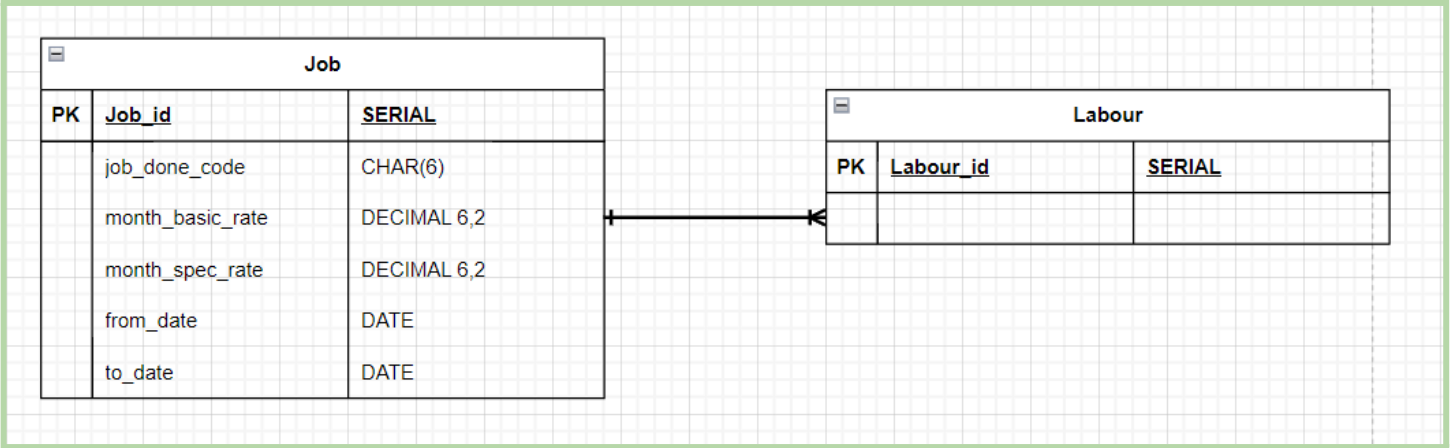
Q7. In a manufacturing industry labourers are given different jobs on different days and each job has its own monthly basic and monthly special rates as wages to be paid to labourers. A worker is not given more than one type of job on a day. A database designer is given the job of designing a database for the above situation and the designer designs a draft of one of the tables as :

FIELD	TYPE	REMARKS
From date	DATE	From this date to
To date	DATE	This date
Labour ID	SERIAL	The worker ID
Job done code	CHAR(6)	The job ID
At Basic Rate	DECIMAL(10,2)	At this basic rate
At Special Rate	DECIMAL(10,2)	At this special rate

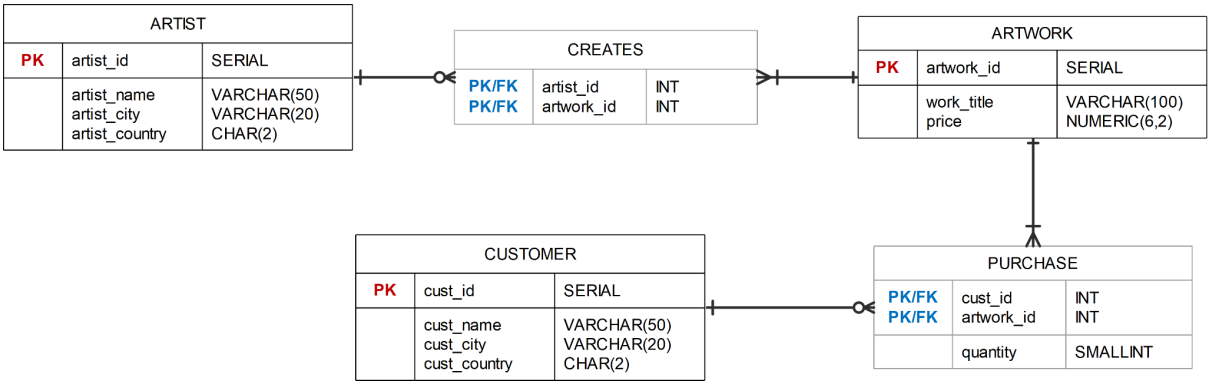
Fig. 3

Draw an ERD for the above situation that represents the relationship between **job** and **labour** (place each attribute in the correct table along with data type/size).

ANSWER:



Create a new database in your VM machine named **lab1** and paste the provided SQL code [dbprin_lab1] into the database. Analysing the provided ERD (Figure 4) and the code, provide answers for the following questions.



Note: *You should format ALL your queries for the optimal output using appropriate column names and CONCAT where available (e.g. instead of "cus_name" it should be "Customer Name").*

Q8. Find the name of the artist who has created the Artwork titled '*Rainbow*'.

```
SELECT
  a.artist_name as "Artist name",
  ar.work_title as "Title"
From
  artist a
  join creates c on c.artist_id = a.artist_id
  join artwork ar on c.artwork_id = ar.artwork_id
where
  work_title = 'Rainbow';
```

```
 Artist name |  Title
-------------+---------
 Ringo       | Rainbow
(1 row)
```

Q9. Find the titles of the Artworks created by '*Lolo*'.

```
SELECT
  a.artist_name as "Artist name",
  ar.work_title as "Title"
From
  artist a
  join creates c on c.artist_id = a.artist_id
  join artwork ar on c.artwork_id = ar.artwork_id
where
  a.artist_name = 'Lolo';
```

```
 Artist name |       Title
-------------+--------------------
 Lolo        | Colours of the Sky
 Lolo        | Long road to home
(2 rows)
```

Q10. Find the names of customers who have not bought an artwork priced more than £ 200. List the customer name, artwork title and the price. Add the £ sign to the output and order on price from lowest to highest.

```
SELECT
  c.cust_name as "Customer name",
  art.work_title as "Title",
  CONCAT_WS (' ', '£', art.price) as "Price"
FROM
  customer c
  join purchase p on p.cust_id = c.cust_id
  join artwork art on p.artwork_id = art.artwork_id
WHERE
  art.price <= '200'
Order by
  art.price ASC;
```

Customer name	Title	Price
Mary	The war	£ 1.00
Michael	Reflection	£ 40.00
Mary	Reflection	£ 40.00
John	Reflection	£ 40.00
Sally	The moon	£ 145.00
Margaret	Night street	£ 150.00
Michael	Long road to home	£ 150.00
Paul	Long road to home	£ 150.00
Sally	Long road to home	£ 150.00
Paul	Night street	£ 150.00
Harry	My lovely song	£ 200.00

(11 rows)

.

Conclusion/Reflection

Questions 5 and 7 were slightly confusing for me, but I finally managed to solve them. I enjoyed the last 3 questions.

LAB 2 (10/10/2023)
Normalization and Denormalization

Lab Description

This lab focuses on the concepts of **normalization** and **denormalization** in database design. Understanding these concepts is essential for building an efficient and flexible database structure that maintains data integrity. In this lab, we will explore how to organize data by carefully considering relationships and minimizing redundancy to prevent data inconsistencies.

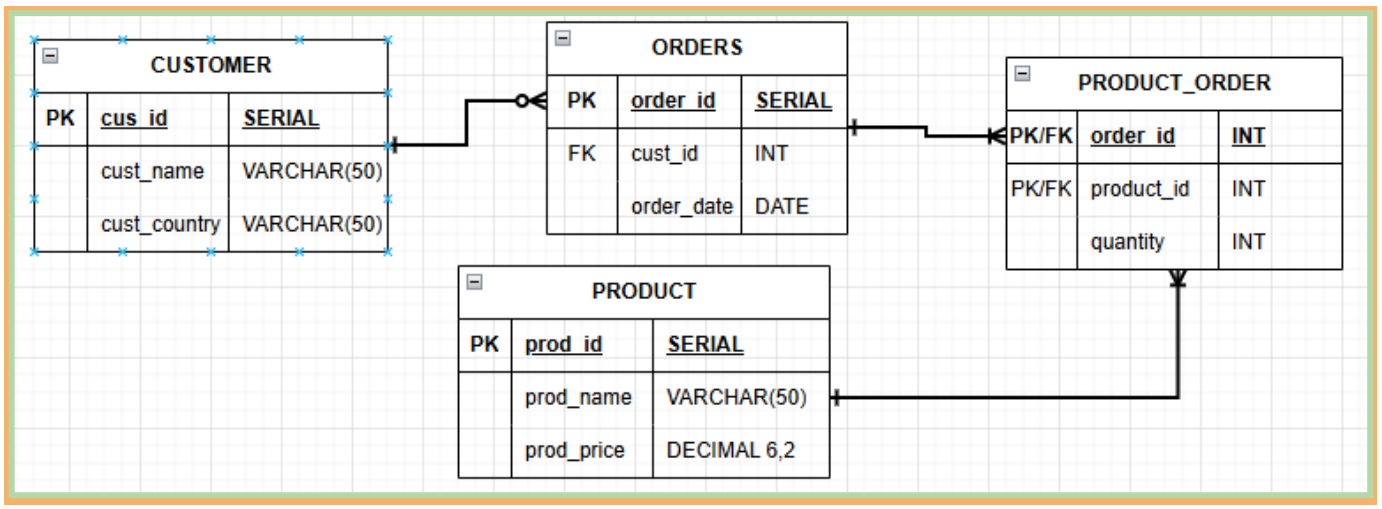
— LAB ANSWERS —

Q1. You have the following table (Fig.1) with data. Normalise the table in 1NF (show 1NF) and create the ERD for 3NF, with the associated data type.

order_id	order_date	cust_id	cust_name	cust_country	prod_id	prod_name	prod_price	prod_qty
1001 1001 1001	01/10/2022 01/10/2022 01/10/2022	1	Apple Apple Apple	US	7, 5, 4	table, desk, chair	800, 325, 200	1, 1, 5
1002 1002	02/10/2022 02/10/2022	2	Samsung	KO	11, 4	dresser, chair	500, 200	4, 2
1003	03/10/2022	3	Benq	DE	11	dresser	500	3

Answer:

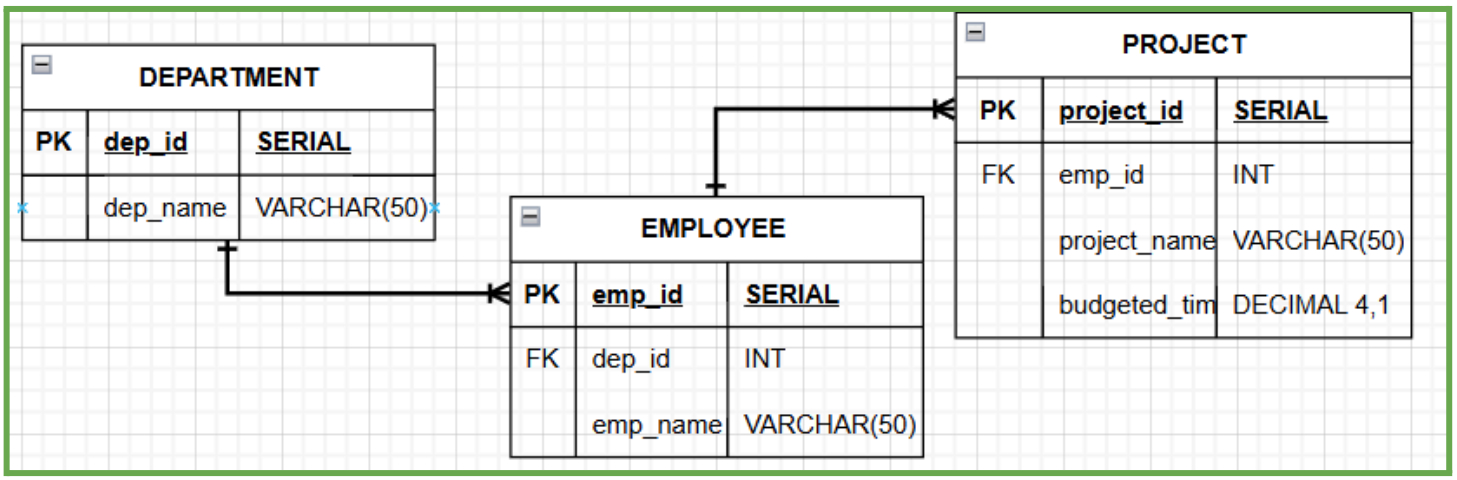
order_id	order_date	cust_id	cust_name	cust_country	prod_id	prod_name	proid_price	prod_qty
1001	01/10/2022	1	Apple	US	7	table	800	1
1001	01/10/2022	1	Apple	US	5	desk	325	1
1001	01/10/2022	1	Apple	US	4	chair	200	5
1002	02/10/2022	2	samsung	KO	11	dresser	500	4
1002	02/10/2022	2	samsung	KO	4	chair	200	2
1003	03/10/2022	3	Benq	DE	11	dresser	500	3



Q2. You are given the following table in 1NF. Normalise data in 3NF and create the ERD with all data types and sizes.

dept_id	dep_name	emp_id	emp_name	project_id	project_name	budgeted_time
10	Finance	1	Harry	100	Alpha	4.5
10	Finance	5	Dewey	105	Beta	3
10	Finance	11	Louie	103	Gamma	7
20	R&D	2	Jack	107	Delta	8
20	R&D	4	Jill	102	Echo	9

ANSWER:



Description of any problem encountered (if any):

this question makes me feel insecure about giving my answer right away. Based on all the examples and knowledge my logic would be that the relation between the Project and the employee will be many to many. In this case, the intersection table between them will be mandatory. But because we don't know the business rules and needs, we only work with the given information. Looking at this table and only the provided information we can assume that one employee will work on one or many projects.

Q3. The University of Portsmouth keeps the following details about a student and the various modules the student studied (not accurate):

up_number (student registration number to university)
stu_name (student name)
stu_addr (student address)
tut_id (tutor id)
tut_name (tutor name)
course_id - (course code)
course_name (course name)
module_id (module cod)
module_name (module name)
module_results (module exam result)

in a relation:
Student(up_number, stu_name, stu_addr, tut_id, tut_name, course_id, course_name, (module_code, module_name, module_results))
The functional dependencies are:

course_code -> course_name
tut_id -> tut_name
up_number, module_id -> module_results
module_code -> module_name

Which of the following is a normalisation of the relation **Students to 1FN**?
Hint: you can input a quick sample output in a spreadsheet (like Q1) and remove the repeating group.

a. **STUDENT(up_number**, stu_name, stu_addr, tutor_id, tutor_name, course_id, course_name)
MODULE(up_number, **module_id**, module_name, module_results)
b. **STUDENT(up_number**, stu_name, stu_addr, tutor_id, tutor_name, course_id, course_name, (module_code, module_results))
MODULE (**module_code**, module_name)
c. **STUDENT(up_number**, stu_name, stu_addr, tutor_id, tutor_name, course_id, (module_id, module_name, module_results))
COURSE(course_id, course_name)
d. **STUDENT(student_id,** stu_name, stu_addr, tutor_id, tutor_name, course_id)
MODULE(student_id, **module_id**, module_name, module_results)
COURSE(course_id, course_name)

Answer:

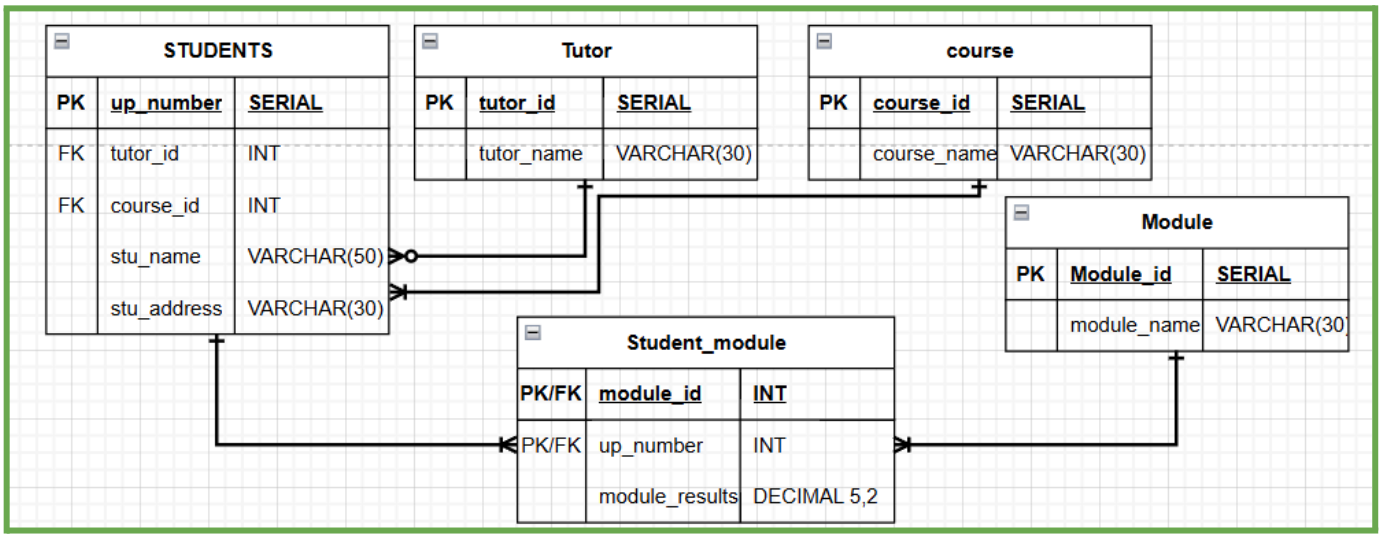
STUDENT(up_number, stu_name, stu_addr, tutor_id, tutor_name, course_id, course_name)
MODULE(up_number, **module_id**, module_name, module_results)

students						
up_number	stu_name	stu_address	tutor_id	tutor_name	course_id	course_name
354354534	Reneta	worhing rd	5	Jhon	2	computing

Modules			
up_number	module_id	mod_name	modu_result
56646	6	networks	60

Q4. Based on the previous data sample, what would be a 3NF normalisation? Write just the table name and the attributes as above

ANSWER:

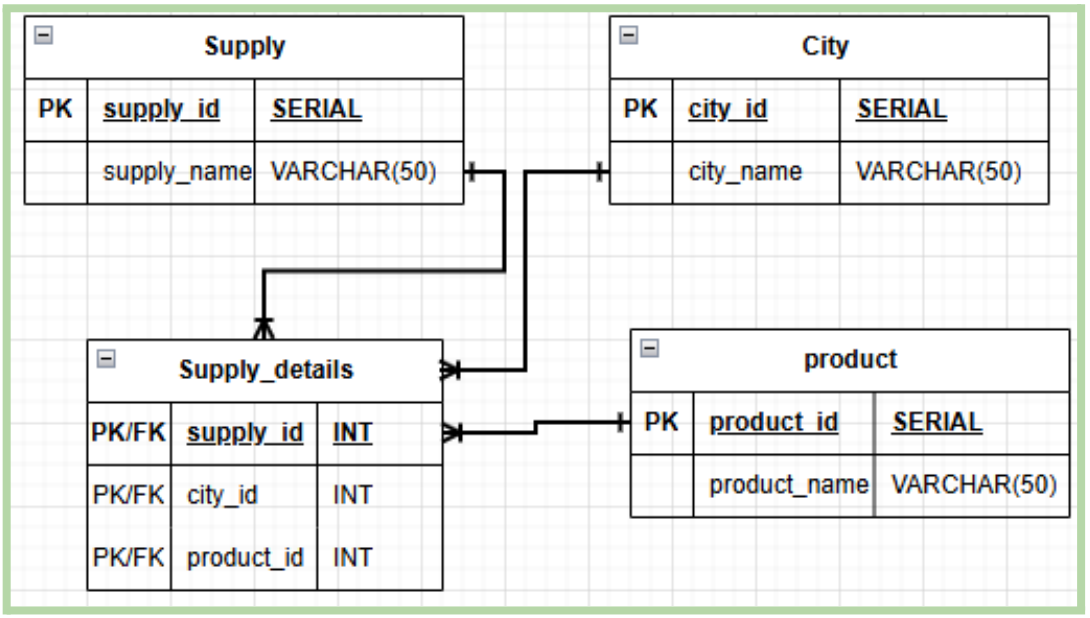


Q5. You have a SUPPLY table that records details about suppliers, products, and cities where the products are supplied. Acknowledge that a supplier will supply in many cities and a city will have many suppliers.

`SUPPLY(supplier_id, supplier_name, product_id, product_name, city_id, city_name)`

Normalize the table into 3NF. Assume that the table is already in 1NF, explain why certain attributes had to be separated.

ANSWER:



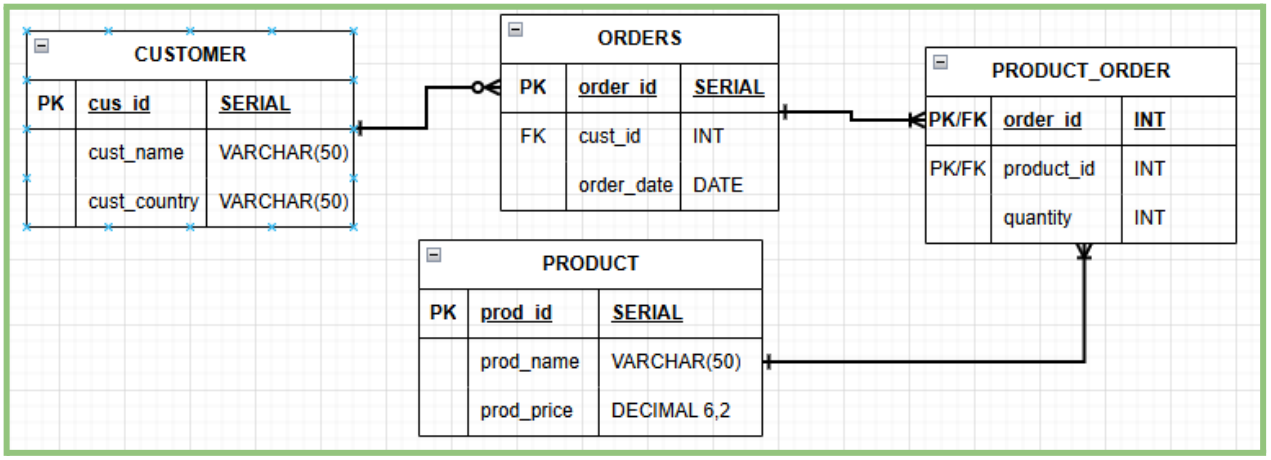
We need to separate the attributes to avoid redundancy. Also, in 3NF, the data needs to be dependent only on the primary key.

Q6. Consider the following ORDER table that records:

`ORDER(order_id, customer_id, customer_name, product_id, product_name, quantity, date)`

Normalize the ORDER table to 3NF.

ANSWER:



Q7. Open your VM and create a new database (lab2). You are given the following data dictionary.

ANIMAL_TYPE						
Atribute Name	**Data Type**	**Size**	**Key**	**Reference**	**Constraints**	**Description**
animal_type_id	SERIAL		**PK**			
common_name	VARCHAR	50			NOT NULL, UNIQUE	(eg. 'Arctic Wolf')
scientific_name	VARCHAR	150			NOT NULL	Official scientificc name of the animal
conservation_status	VARCHAR	50			NOT NULL	('Endagered', 'Least Concerned')

MENAGERIE						
Atribute Name	Data Type	Size	Key	Reference	Constraints	Description
menagerie_id	SERIAL		**PK**			
common_name	VARCHAR	50	**FK**	animal_type > common_name	NOT NULL	
date_aquired	DATE				NOT NULL	
gender	CHAR	1			NOT NULL	
aquired_from	VARCHAR	250			NOT NULL	
name	VARCHAR	50			NOT NULL	
notes	TEXT					

…and the data sample output:

animal_type_id	common_name	scientific_name	conservation_status
1	Bengal Tiger	Panthera tigris tigris	Endangered
2	Arctic Wolf	Canis lupus arctos	Least Concern

menagerie_id	common_name	date_acquired	gender	acquired_from	name	notes
1	Bengal Tiger	2011-07-14	F	Dhaka Zoo	Ariel	Healthy coat at last exam.
2	Arctic Wolf	2008-09-30	F	National Zoo	Freddy	Strong appetite.
3	Bengal Tiger	2006-06-01	M	Scotland Zoo	Spark	Likes to play
4	Arctic Wolf	2007-06-12	F	Southampton National Park	Mia	Doesn't like sun

- Write the CREATE statements for tables ANIMAL_TYPE and MENAGERIE, including PKs, FKs, constraints, data type and size.
- Write 6 INSERT STATEMENTS for the following:
  - Common Name: ‘Bengal Tiger’, ‘Arctic Wolf’
  - Scientific Name: ‘Panthera tigris tigris’, ‘Canis lupus arctos’
  - Conservation Status: ‘Endangered’, ‘Least Concern’
  - Acquired Date: ‘14/07/2011’, ‘30/09/2008’, ‘01/06/2006’, ‘12/06/2007’
  - Gender: ‘M’, ‘F’
  - Acquired From: ‘Dhaka Zoo’, ‘National Zoo’, ‘Scotland Zoo’, ‘Southampton National Park’
  - Name: ‘Ariel’, ‘Freddy’, ‘Spark’, ‘Mia’
  - Notes: ‘Healthy coat at last exam’, ‘Strong appetite’, ‘Likes to play’, ‘Doesn't like sun’

The output of both tables should be exactly like in provided examples for ANIMAL_TYPE and MENAGERIE .

Q8. Based on the previous database you have created, list all the animals that are endangered, along with common name, scientific name, animal name and date acquired.

```
--CREATE DATABASE animal;
```

```
SELECT
  at.common_name as "Common Name",
  at.scientific_name as "Scientific name",
  m.name as "Animal name",
  m.date_aquired as "Data of aquired"
From
  animal_type at
  JOIN menagerie m on at.common_name = m.common_name
WHERE
  conservation_status = 'Endangered';
```

```
 Common Name  |    Scientific name     | Animal name | Data of aquired
--------------+------------------------+-------------+-----------------
 Bengal Tiger | Panthera tigris tigris | Ariel       | 2011-07-14
 Bengal Tiger | Panthera tigris tigris | Spark       | 2006-06-01
(2 rows)
```

.

Conclusion/Reflection

Normalisation and denormalisation always was slightly confusing for me but with this lab, I think started feeling more confident in this important process.

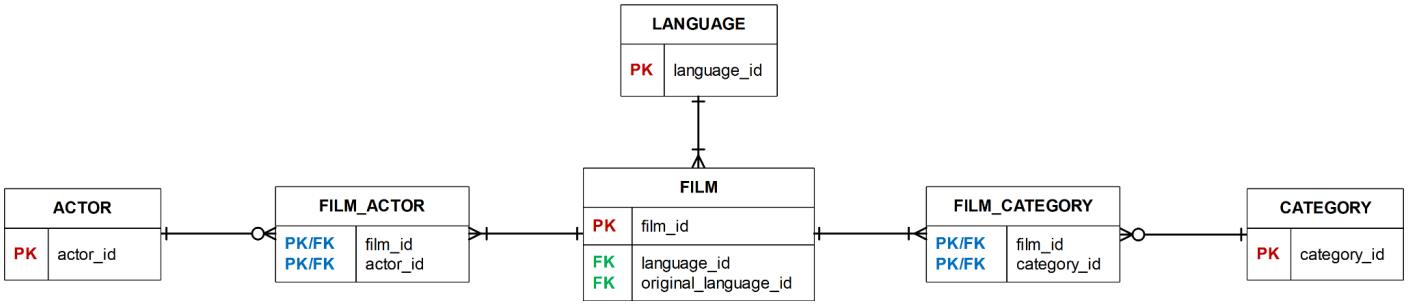
LAB 3 (17/10/2023)
Transactions

Lab Description

A very short description of your own understanding of the lab requirements. You do not have a word/page limit but keep it sensible - we do not need essays.

Q1. You are given the following ERD without any additional attributes. Build up your database using your VM and **write 4 separate queries** that will contain a **SELECT**, **INSERT, UPDATE** and **DELETE** that will query the database. Create the ***Transaction Analysis Matrix*** for those transactions.

Do not spend time making the matrix "pretty" but accurate. You can use a spreadsheet or a table inserted in a document. Examples of the matrix are in the lecture presentation. Don't forget to add your work to the logbook.



(Query example - You should add some attributes to make the database functional (e.g. actor_name; actor_lname etc)

List the name of all movies where the actor's id (or surname) is X.

```
SELECT
  actor_name, film_name
FROM
  table_name
  JOIN table_name ON table1.attr = table2.attr
  JOIN table_name ON table3.attr = table4.attr
WHERE
  table.attr = 'X';
```

Same for a new INSERT and for an UPDATE

ANSWER:

SELECT

```
--list all movies that actor with id -5 Participate.
Select
  CONCAT_WS (
    ' ',
    a.actor_first_name,
    a.actor_middle_name,
    a.actor_last_name
  ) as "Actor name",
  f.film_name as "Movie"
from
  actor a
  join film_actor fa on fa.actor_id = a.actor_id
  join film f on f.film_id = fa.film_id
WHERE
  a.actor_id = 5;

---in this query Transaction analysis matrix will only read the date
--from table actor, film_actor and film to show us the actor with id5 in which movie
participates.

/* Actor name   |  Movie
--------------+---------
 Meryl Streep | Movie 1
 Meryl Streep | Movie 5
(2 rows)*/
```

INSERT

```
--insert a new author, in this case, the database will first read the rules and constraints
and if everything is fine will create a new entity.
INSERT INTO
  actor(actor_id,
    actor_first_name,
    actor_middle_name,
    actor_last_name
  )
VALUES(7,'Reneta','Stoyanova','Todorova');

/* actor_id | actor_first_name | actor_middle_name | actor_last_name
----------+------------------+-------------------+-----------------
        1 | Robert           |                   | De Niro
        2 | Marlon           |                   | Brando
        3 | Denzel           | Hayes             | Washington
```

```
       4 | Humphrey         | DeForest          | Bogart
       5 | Meryl            |                   | Streep
       6 | Danie            | Michael           | Blake Day-Lewis
       7 | Reneta           | Stoyanova         | Todorova
(7 rows)*/
```

UPDATE

```
--UPDATE  in film with id=1 replace actor with id = 6 with the new actor added with id=7
UPDATE film_actor
SET actor_id = 7
WHERE film_id = 1 and actor_id= 6;

select* from film_actor;-- before UPDATE
/*film_id | actor_id
---------+----------
       1 |        6
       1 |        5
       2 |        4
       2 |        1
       3 |        2
       3 |        4
       4 |        2
       4 |        3
       5 |        6
       5 |        5
(10 rows)*/

-- after UPDATE STATEMENT

/* film_id | actor_id
---------+----------
       1 |        5
       2 |        4
       2 |        1
       3 |        2
       3 |        4
       4 |        2
       4 |        3
       5 |        6
       5 |        5
       1 |        7
(10 rows)*/
```

```
-- If we try to update with an ID that doesn`t exist nothing will happen
```

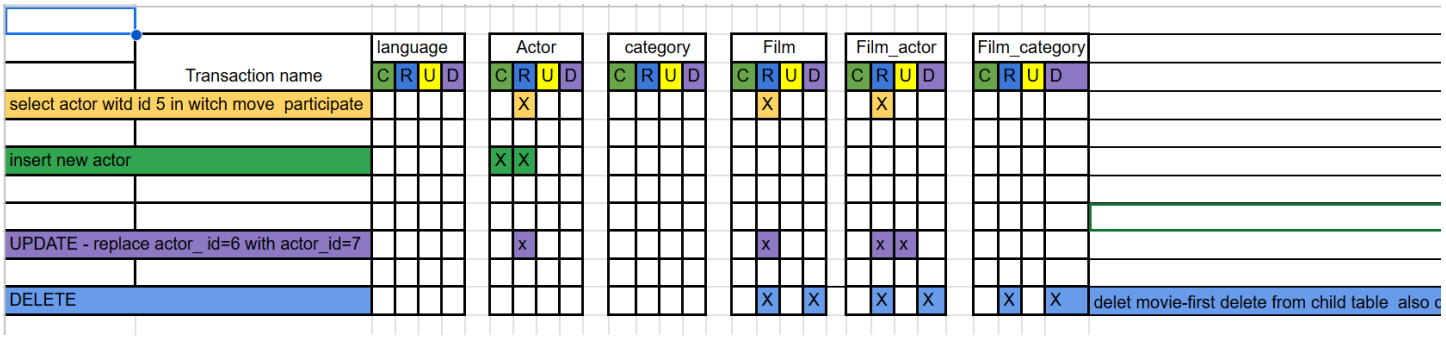
DELETE

```
-delete movie with id=5
--delete first from the child table

select * from film_actor; --before delete

/* film_id | actor_id
---------+----------
       1 |        5
       2 |        4
       2 |        1
       3 |        2
       3 |        4
       4 |        2
       4 |        3
       5 |        6
       5 |        5
       1 |        7
(10 rows)*/

DELETE FROM film_actor
WHERE film_id=5;
                        --      after delete

                            /* film_id | actor_id
                            ---------+----------
                                  1 |        5
                                  2 |        4
                                  2 |        1
                                  3 |        2
                                  3 |        4
                                  4 |        2
                                  4 |        3
                                  1 |        7
                            (8 rows)*/


--before delete
select * from film_category;
 /*film_id | cat_id
```

```
---------+--------
       1 |      3
       2 |      4
       3 |      1
       4 |      2
       5 |      3
(5 rows)*/

DELETE FROM film_category
WHERE film_id=5;

               ---after delete
               /*film_id | cat_id
               ---------+--------
                     1 |      3
                     2 |      4
                     3 |      1
                     4 |      2
               (4 rows)*/


select* from film;
/*
 film_id | lang_id | original_lang_id | film_name
---------+---------+------------------+-----------
       1 |       1 |                3 | Movie 1
       2 |       1 |                2 | Movie 2
       3 |       2 |                3 | Movie 3
       4 |       4 |                1 | Movie 4
       5 |       4 |                3 | Movie 5
(5 rows)*/

DELETE FROM film
WHERE film_id=5;
--after delete
/* film_id | lang_id | original_lang_id | film_name
---------+---------+------------------+-----------
       1 |       1 |                3 | Movie 1
       2 |       1 |                2 | Movie 2
       3 |       2 |                3 | Movie 3
       4 |       4 |                1 | Movie 4
(4 rows)*/
```

```
DELETE FROM film
WHERE film_id=1;
/*update or delete on the table "film" violates foreign key constraint
"film_actor_film_id_fkey" on table "film_actor"
DETAIL:  Key (film_id)=(1) is still referenced from table "film_actor".*/
```

My answer can be found here crud TA MATRIX



Transaction name	language				Actor				category				Film				Film_actor				Film_category				
	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	
select actor witd id 5 in witch move participate						X								X				X							
insert new actor					X	X																			
UPDATE - replace actor_ id=6 with actor_id=7						x								x				x	x						
DELETE														X		X		X		X		X		X	delet movie-first delete from child table also c

Q2. Define in your own understanding (do not copy/paste definitions) of these terms:

- **Atomicity-** The transaction needs to be finished successfully if any part fail none of the changes will be saved
- **Consistency-** after committing the transaction all data must meet all the rules and constraints
- **Isolation-** each transaction is independent of any other transactions
- **Durability** changes will be saved after the transaction is finished even if the system fails
- **Schedule** - is an approach for managing a concurrency control, don`t allow two transactions to interfere with each other
- **blind write** - refers to writing operations in the database without prior reading
- **dirty read** - when one person can see the other person`s work before being committed ()
- **unrepeatable read-** when two different values of the same data are read in the same transaction because another transaction changes the value between reads.
- **serialisable schedule -** when two transactions are executed concurrently but always leave the database in a consistent state.
- **recoverable schedule-** A recoverable schedule is when one transaction is committed only if all transactions that they depend on are successfully committed
- **avoids-cascading-aborts schedule** - this schedule prevents dirty reading problems by not allowing data reading from uncommitted transactions and prevents the rollback of dependent transactions, which could lead to information integrity violations.
-

Note: *Some of the answers were in the lecture, some in the book.*

Q3. Consider the situation below, in which a number of account records have the following values:***A1 = £40; A2 = £50; A3 = £30***

To transfer £10 from A3 to A1 while concurrently calculating the total funds in the three accounts, the following sequence of events may occur. Show the value of each data item in the last column, and discuss the reason for an incorrect summary value.

Note: Look at the lecture example and/or chapter 22 in Connolly and Begg.

ANSWER: My answer can be found here: Concurrency transactions

Time	Transaction1	Transaction2	SUM
t_1	sum = 0		
t_2	read(**A1**)		
t_3	sum = sum + A1		
t4		read(**A3**)	
t5		A3 = A3 - 10	
t6		write(A3)	
t7		read(A1)	
t8		A1 = A1 + 10	
t9		write(A1)	
t10		commit;	
t11	read(**A3**)		
t12	sum = sum + A3		
t13	read(**A2**)		
t14	sum = sum + A2		

Answer with a screenshot below:

	Person 1	Person 2		£40.00	£50.00	£30.00	
Time	**Transaction1**	**Transaction2**	**SUM**	**Account 1**	**Account 2**	**Account 3**	
t1	sum = 0		0	**£40.00**	**£50.00**	**£30.00**	Person 1 neet to calculate the total funds in all 3 accounts
t2	read(A1)			**£40.00**			
t3	sum = sum + A1		**£40.00**				the value for the -->sum=0+40 so the new value of sum is 40
t4		read(A3)				**£30.00**	
t5		A3 = A3 - 10					30-10=20 person 2 start working and start transfering 10 pound from A3 to A1
t6		write(A3)				**£20.00**	
t7		read(A1)		**£40.00**			
t8		A1 = A1 + 10					40+10=50
t9		write(A1)		**£50.00**			
t10		commit;		**£50.00**	**£50.00**	**£20.00**	job done- transaction commited the new values are as follows.The sum of all accounts need to be = £120
t11	read(A3)					**£20.00**	
t12	sum = sum + A3		**£60.00**				new sum = 40-> then for **t12 sum = 40+20 and we have new value of the sum of 60**
t13	read(A2)				**£50.00**		
t14	sum = sum + A2		£110.00				last sum is 60 then 60+50=110

To transfer £10 from A3 to A1 while concurrently calculating the total funds in the three accounts, the following sequence of events may occur. Show the value of each data item in the last column, and discuss the reason for an incorrect summary value.

The reason for the incorrect summary value is because both transaction work conccurently, and person 1 read A1 but later the amount is change from person 2 . and before person 1 to finish the transaction person 2 make transfer witch lead to wrong summary value.

.

Conclusion/Reflection

The most difficult part of this lab was to explain all definitions in my own words.

LAB 4 (24/10/2024)
Relational Algebra

— LAB ANSWERS —
The purpose of this lab is to give us hands-on experience with relational algebra witch provides a formal way to express queries using mathematical expressions. We`ll focus on breaking down queries step-by-step following a key rule - query should be executed inside to outside.

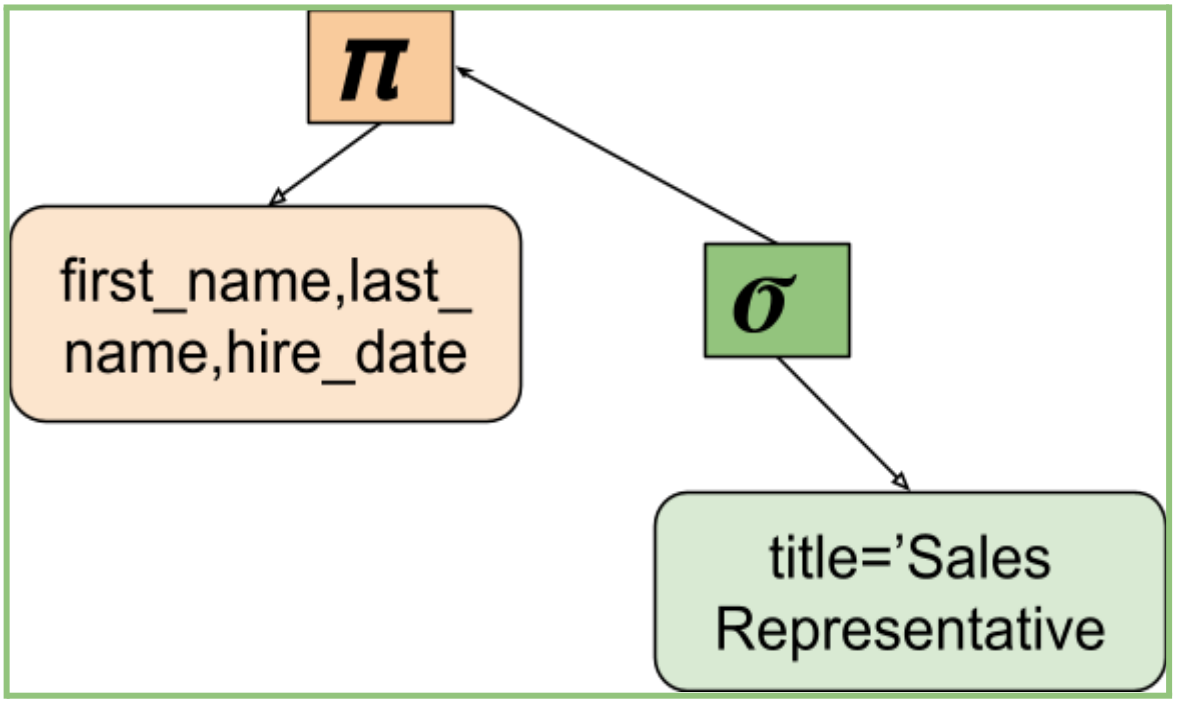
Q1 .You are given the following query:

```
SELECT first_name, last_name, hire_date
FROM Employees
WHERE
      Title = 'Sales Representative';
```

Write the correspondent RA with π and σ and draw the tree diagram to show this relation

Answer Q1:

π first_name,last_name,hire_date$(\sigma$ title=' Sales Representative' (Employees)$)$

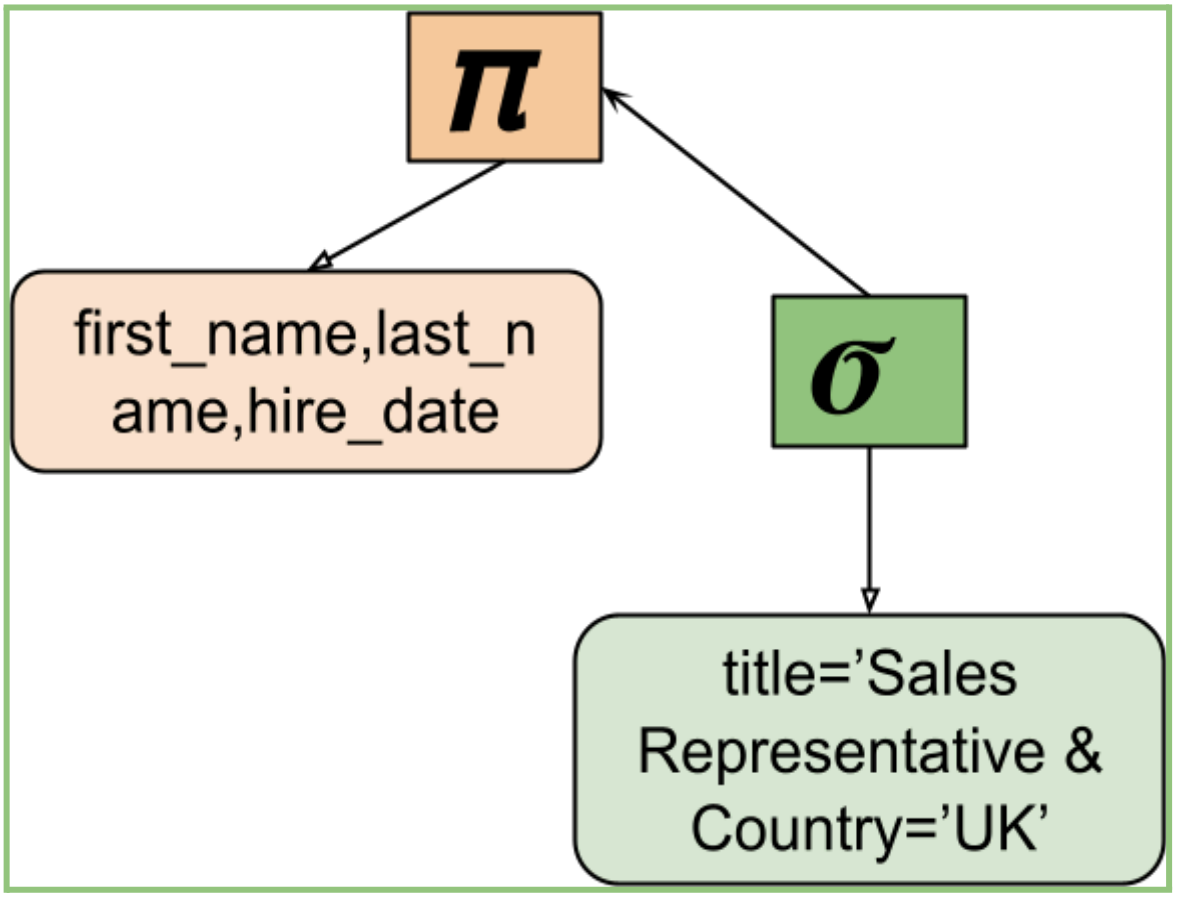


Q1. Let's expand the above query and we are adding a second WHERE clause filter. Write the new RA and the tree diagram.

```
Select first_name, last_name, hire_date
From Employees
Where
      Title = 'Sales Representative'
      and Country = 'UK';
```

Answer Q2:

π first_name,last_name,hire_date$(\sigma$ title=' Sales Representative' & Country= 'UK' (Employees)$)$

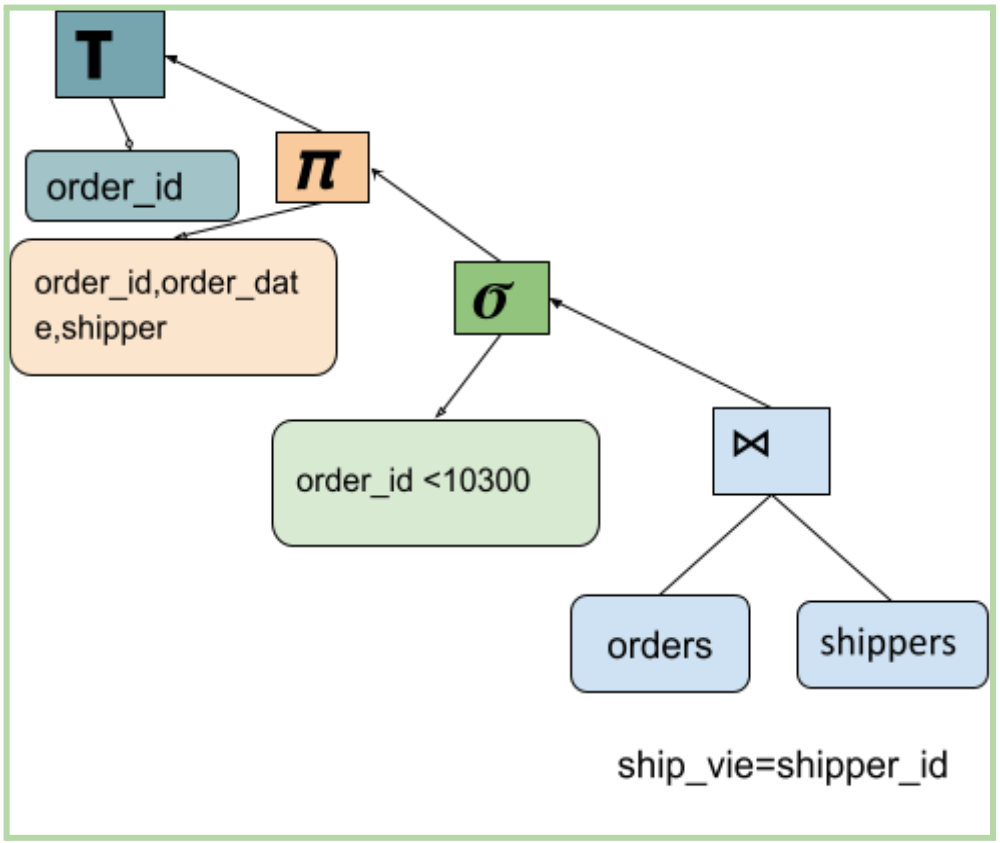


Q2. The following query has a conditional WHERE clause and ordering. What would be the RA for this?

```
SELECT order_id, order_date, shipper
FROM orders INNER JOIN shippers ON shippers . shipper_id = orders . ship_via
WHERE order_id < 10300
ORDER BY order_id
```

Answer Q3:

$$\mathbf{T}_{\text{order\_id}}\,\pi_{\text{order\_id,order\_date,shipper}}\left(\sigma_{\text{order\_id}<10300}\left(\text{Orders}\bowtie_{\text{ship.vie=shipper\_id}}\text{shippers}\right)\right)$$



Q3. Consider a database with the following schema:

USER	(name, age, gender) PK - name
FREQUENTS	(name, pizzeria) PK name, pizzeria
EATS	(name, pizza) PK - name, pizza
SERVES	(pizzeria, pizza, price) PK - pizzeria, pizza

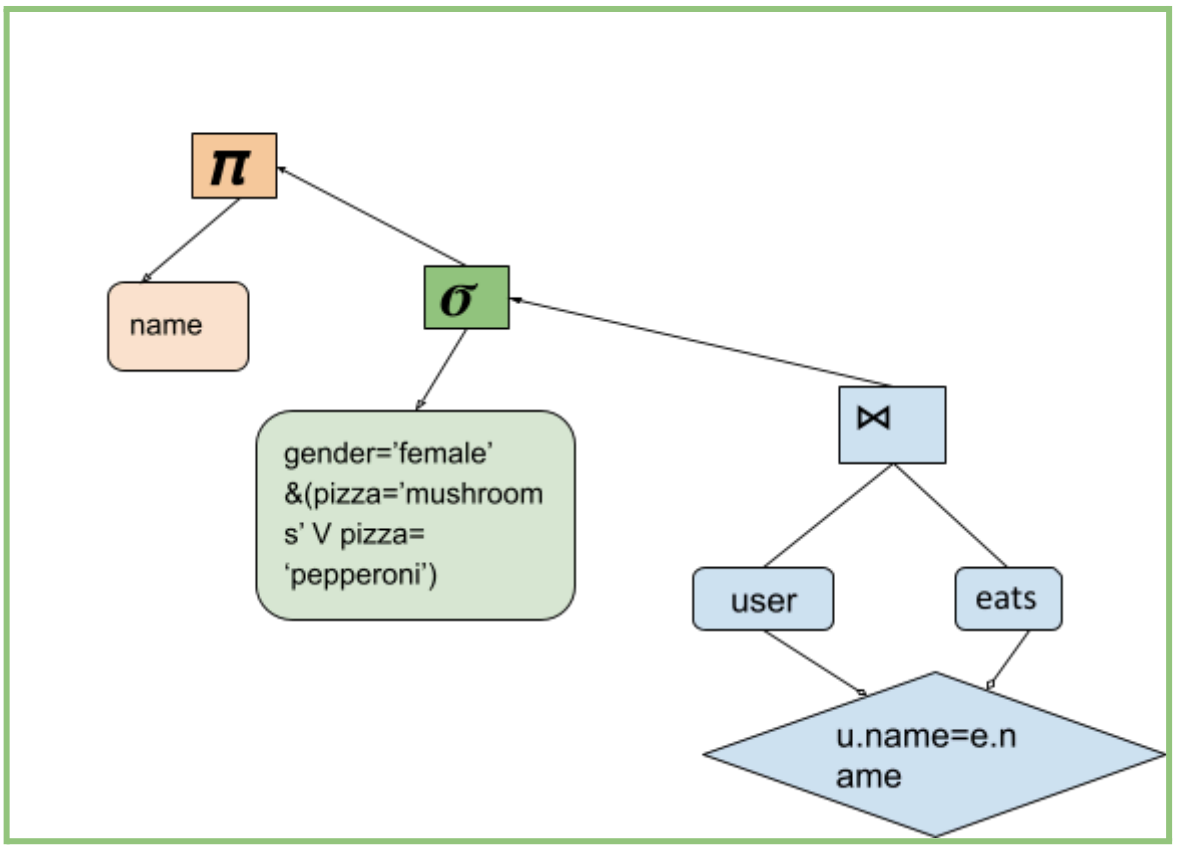
Write the relational algebra expressions and query tree for the following output.

"*Find the names of all females who eat either mushroom or pepperoni pizza (or both).*"

Hint: *To visualise the above schema more easily, you can quickly draw the ERD using pen and paper.*

ANSWER:

$$\pi_{\text{name}}\left(\sigma_{\text{gender= 'female' and(pizza='mushroom' or pizza='pepperoni ')}}\left(\text{USER} \bowtie_{\text{u.name=e.name}} \text{EATS}\right)\right)$$



Q4. Consider a database with the following schema:

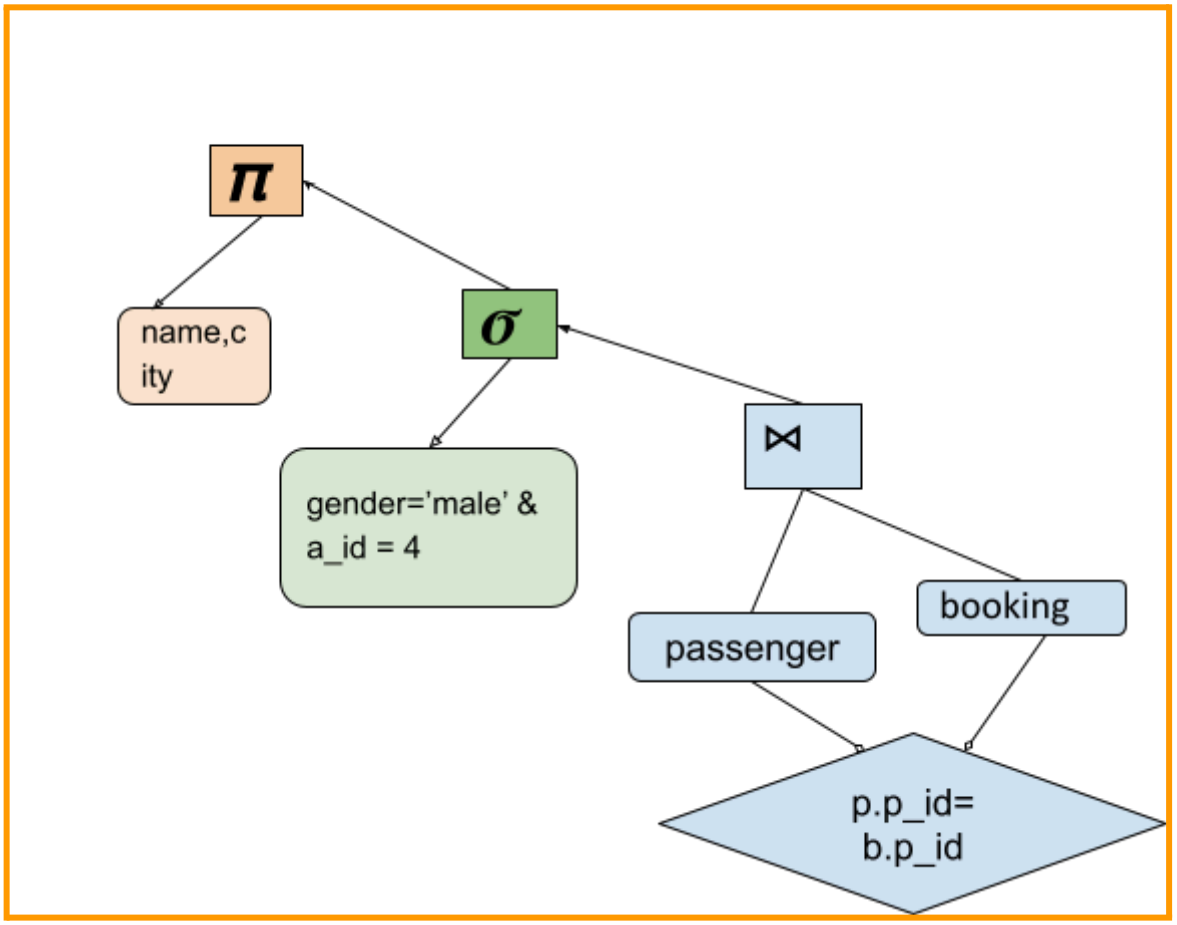
PASSENGER	(**p_id,** p_name, p_gender, p_city)
AGENCY	(**a_id**, a_name, a_city)

FLIGHT (**f_id**, f_date, time, src, dest)
BOOKING (**p_id***, **a_id***, **f_id***, **f_date**)

Write the relational algebra expression and query tree for the following request:
"*List details of all male passengers who are booked through this agency*"

ANSWER:

$$\pi_{p.name,p.city}(\sigma_{gender='male' \,\&\, a\_id='4'}(PASSENGER \bowtie_{b.p\_id=p.p\_id} BOOKING))$$



Q5. Relational algebra is a _____ Data Manipulation Language (DML).

a. **Declarative**
b. **Object-oriented**

c. **Procedural**
d. **Static**
e. **None of the above**

Q6. Assuming that we have two relations R_1 and R_2 where R_1 has 10 tuples (records) and R_2 5, how many records will be there in the result of the operation R_1 x R_2 (cartesian product)?

a. **10 Records**
b. **5 Records**
c. **15 Records**
d. **50 Records**
e. **None**

Q7. For relation *Student(up, name, school)*, suppose school can be one of {soc, smp, sces, smde, seee, dlw, icg}.
If the number of tuples in relation to Student is 1000, then what is the size estimate of the query $\sigma_{\text{school = '}physics\text{'}}$(Student)?

ANSWER Q8: we don`t have a school named *'physics'* so the query will return 0.

Conclusion/Reflection

I encountered some difficulties in questions 4 and 5, but after a second reading of the task and assumptions, I finally solved the task.

LAB 5
Optimizations

Q1. The below SQL statement has five syntax errors. Can you identify them? What would be the correct statement to return *exactly the same result* as in Fig1? Write the correct SQL statement.

```
SELECT
  movie_id AS "Movie ID",
  movie_title AS Movie_Title
  movie_lang AS 'LANGUAGE',
  cat_name AS "Category"
FROM
  movie
  INNER JOIN category LINK movie.movie_id = category.movie_id
WHERE
  movie_title = ALADDIN CALENDAR
```

```
 Movie ID |   movie_title    |       language       | Category
----------+------------------+----------------------+----------
       10 | ALADDIN CALENDAR | English              | Sports
(1 row)
```

ANSWER:

```
SELECT
  m.movie_id AS "Movie ID",
  m.movie_title AS "Movie_Title",
  m.movie_lang AS "LANGUAGE",
  c.cat_name AS "Category"
FROM
  movie m
  INNER JOIN category c ON m.movie_id = c.movie_id
WHERE
  m.movie_title = 'ALADDIN CALENDAR';
```

Q2. You are given the following two queries. The table employee have numerous attributes and your department needs the name, NIN and hire date of an employee. Which one will have a better performance and why?

A

```
SELECT
  *
FROM
  employee
WHERE
  employee_id = 1;
```

B

```
SELECT
  emp_name,
  emp_last_name,
  emp_nin,
  emp_hire_date
from
  employee
WHERE
  employee_id -1;
```

ANSWER:
It is always a better solution to specify the attribute that you need instead of using SELECT *. However, in this case, query A will perform better because query B has a mistake and returns an error. WHERE clause should be written employee_id = 1, not -1.

The full ERD and an Interactive Data Dictionary.

Q3. Using DB-specific commands, How many ***VIEWS*** are inside of the movie_rental database?

Answer: This database has 7 view tables. The comment to receive this information is \dv

```
 Schema |            Name             | Type |  Owner
--------+-----------------------------+------+----------
 public | actor_info                  | view | up2246775
 public | customer_list               | view | up2246775
 public | movie_list                  | view | up2246775
 public | nicer_but_slower_movie_list | view | up2246775
 public | sales_by_movie_category     | view | up2246775
 public | sales_by_store              | view | up2246775
 public | staff_list                  | view | up2246775
(7 rows)
```

Q4. Using the provided ERD and Data Dictionary, create a query that will return all the movie titles that are in the ***Documentary*** category. Create 2 versions of the same query and in one use category as

string (‘**documentary’)** and in the other use category ID (you will need to check what is the category ID for **Documentary**). Use the **EXPLAIN ANALYZE** command and see which query performs better? Why?

Note: More information about analysing queries you can find here.

ANSWER : `EXPLAIN ANALYZE`

```
SELECT
    m.title as "Movie Title"
FROM
    movie m
    JOIN movie_category mc on mc.movie_id = m.movie_id
    JOIN category c on c.category_id = mc.category_id
where
    c.name = 'Documentary';
```

```
/*                                    QUERY PLAN

-------------------------------------------------------------------------------------------------
---------------------------------
 Nested Loop  (cost=16.69..38.26 rows=6 width=15) (actual time=0.037..0.460 rows=68 loops=1)
 ->  Hash Join  (cost=16.41..35.06 rows=6 width=4) (actual time=0.031..0.296 rows=68 loops=1)
 Hash Cond: (mc.category_id = c.category_id)
  ->    Seq  Scan  on  movie_category  mc    (cost=0.00..16.00  rows=1000  width=8)  (actual
time=0.010..0.167 rows=1000 loops=1)
 ->  Hash  (cost=16.38..16.38 rows=3 width=4) (actual time=0.013..0.014 rows=1 loops=1)
 Buckets: 1024  Batches: 1  Memory Usage: 9kB
  ->   Seq Scan on category c   (cost=0.00..16.38 rows=3 width=4) (actual time=0.009..0.010
rows=1 loops=1)
 Filter: ((name)::text = 'Documentary'::text)
 Rows Removed by Filter: 15
  ->   Index Scan using movie_pkey on movie m   (cost=0.28..0.53 rows=1 width=19) (actual
time=0.002..0.002 rows=1 loops=68)
 Index Cond: (movie_id = mc.movie_id)
 Planning Time: 0.321 ms
 Execution Time: 0.504 ms
 (13 rows)*/
```

```
SELECT
m.title as "Movie Title"
FROM movie m
JOIN movie_category mc on mc.movie_id= m.movie_id
where category_id = 6;
```

```
where category_id = 6;
                                   QUERY PLAN
----------------------------------------------------------------------------------------------------------
 Hash Join  (cost=19.35..88.78 rows=68 width=15) (actual time=0.140..0.430 rows=68 loops=1)
   Hash Cond: (m.movie_id = mc.movie_id)
   ->  Seq Scan on movie m  (cost=0.00..65.00 rows=1000 width=19) (actual time=0.007..0.191 rows=1000 loops=1)
   ->  Hash  (cost=18.50..18.50 rows=68 width=4) (actual time=0.114..0.114 rows=68 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 11kB
         ->  Seq Scan on movie_category mc  (cost=0.00..18.50 rows=68 width=4) (actual time=0.012..0.101 rows=68 loops=1)
               Filter: (category_id = 6)
               Rows Removed by Filter: 932
 Planning Time: 0.246 ms
 Execution Time: 0.460 ms
(10 rows)
```

Q5. Using your SQL knowledge create a query that will list all movies that have original language Italian along with the release year. The movies should be listed by release year in chronological order.

ANSWER:

```
SELECT
m.title AS "Movie title",
m.release_year as "Year"
From movie m
where m.original_language_id = 2
order by m.release_year ASC ;
```

UP2246775_NEW_ DBPRIN_Lab_ 153 books

```
 SPEAKEASY DATE          | 2010
 POTTER CONNECTICUT      | 2010
 WHISPERER GIANT         | 2010
 HOLLOW JEOPARDY         | 2010
 LOVER TRUMAN            | 2010
 OZ LIAISONS             | 2011
 EGG IGBY                | 2012
 CITIZEN SHREK           | 2012
 FERRIS MOTHER           | 2012
 BRANNIGAN SUNRISE       | 2013
 SLUMS DUCK              | 2014
 PLATOON INSTINCT        | 2015
 SHANE DARKNESS          | 2015
 TRAFFIC HOBBIT          | 2015
 CIDER DESIRE            | 2015
 REMEMBER DIARY          | 2015
 ELEMENT FREDDY          | 2015
 OCTOBER SUBMARINE       | 2016
 DESERT POSEIDON         | 2016
 RAGING AIRPLANE         | 2016
 MEMENTO ZOOLANDER       | 2017
 SEA VIRGIN              | 2017
 FLIGHT LIES             | 2018
 ALTER VICTORY           | 2018
 WILD APOLLO             | 2019
 GRIT CLOCKWORK          | 2020
 DOCTOR GRAIL            | 2021
 HIGHBALL POTTER         | 2021
 HARPER DYING            | 2021
 KENTUCKIAN GIANT        | 2022
(153 rows)
```

Q6. List how many movies are in each category per original language. The output should be similar to the one below.

ANSWER:

```
SELECT
c.name AS "Category",
l.name as "original language",
count(m.movie_id) as" number of movies"
From language l
join movie m on l.language_id = m.original_language_id
join movie_category mc on mc.movie_id= m.movie_id
join category c on c.category_id =mc.category_id
```

```
group by c.name,l.name
order by c.name ASC;
```

Category	orifinal language	number of movie
Action	English	7
Action	French	17
Action	German	13
Action	Italian	5
Action	Japanese	6
Action	Mandarin	16
Animation	English	10
Animation	French	14
Animation	German	7
Animation	Italian	11
Animation	Japanese	14
Animation	Mandarin	10
Children	English	12
Children	French	9

Category	Original Language	Number of Movies
Action	English	7
Action	French	17
Action	German	13
Action	Italian	5
Action	Japanese	6
Action	Mandarin	16
Animation	English	10
Animation	French	14
Animation	German	7
Animation	Italian	11

LAB 6

The full ERD and an Interactive Data Dictionary.

Q1. **Connect to the movie_rental database (that you have installed in LAB5) and change the phone number from the current one to *02392844444* for address with id 100. Do not alter any other data/attribute.**

Q2. **Create a query that will return all the details for the record where the phone number is *02392844444* and use EXPLAIN. Take a screenshot of the output.**

ANSWER:

```
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Seq Scan on address a  (cost=0.00..15.54 rows=1 width=63) (actual time=0.081..0.081 rows=1 loops=1)
   Filter: ((phone)::text = '02392844444'::text)
   Rows Removed by Filter: 602
 Planning Time: 0.068 ms
 Execution Time: 0.094 ms
(5 rows)
```

Q3. **Apply an INDEX on the phone attribute.**

```
public | index_phone                                    | index          | up2246775 | address
```

Q4. **Use EXPLAIN again. What are the differences? Which one performs better and why?**

```
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Index Scan using index_phone on address a  (cost=0.28..8.29 rows=1 width=63) (actual time=0.030..0.030 rows=1 loops=1)
   Index Cond: ((phone)::text = '02392844444'::text)
 Planning Time: 0.250 ms
 Execution Time: 0.051 ms
(4 rows)
```

Q5. **Delete the INDEX you have created in Q3**

```
movie_rental=# DROP INDEX index_phone ;
DROP INDEX
```

Q6. Create a VIEW (*customer_details*) that will hold the customers details exactly as below.

Customer	Contact Details	Customer Address	Customer City	Customer Country
VERA MCCOY	VERA.MCCOY@sakilacustomer.org	886649065861	1168 Najafabad Parkway	Kabul	Afghanistan
MARIO CHEATHAM	MARIO.CHEATHAM@sakilacustomer.org	406784385440	1924 Shimonoseki Drive	Batna	Algeria
JUDY GRAY	JUDY.GRAY@sakilacustomer.org	107137400143	1031 Daugavpils Parkway	Bchar	Algeria
JUNE CARROLL	JUNE.CARROLL@sakilacustomer.org	506134035434	757 Rustenburg Avenue	Skikda	Algeria
ANTHONY SCHWAB	ANTHONY.SCHWAB@sakilacustomer.org	478229987054	1892 Nabereznyje Telny Lane	Tafuna	American Samoa
CLAUDE HERZOG	CLAUDE.HERZOG@sakilacustomer.org	105882218332	486 Ondo Parkway	Benguela	Angola
MARTIN BALES	MARTIN.BALES@sakilacustomer.org	106439158941	368 Hunuco Boulevard	Namibe	Angola
BOBBY BOUDREAU	BOBBY.BOUDREAU@sakilacustomer.org	934352415130	1368 Maracabo Boulevard	South Hill	Anguilla
JASON MORRISSEY	JASON.MORRISSEY@sakilacustomer.org	972574862516	1427 A Corua (La Corua) Place	Baha Blanca	Argentina
PERRY SWAFFORD	PERRY.SWAFFORD@sakilacustomer.org	914466027044	773 Dallas Manor	Quilmes	Argentina
DARRYL ASHCRAFT	DARRYL.ASHCRAFT@sakilacustomer.org	717566026669	166 Jinchang Street	Ezeiza	Argentina
MICHEAL FORMAN	MICHEAL.FORMAN@sakilacustomer.org	411549550611	203 Tambaram Street	Escobar	Argentina
JULIA FLORES	JULIA.FLORES@sakilacustomer.org	846225459260	1926 El Alto Avenue	La Plata	Argentina
ERIC ROBERT	ERIC.ROBERT@sakilacustomer.org	105470691550	430 Kumbakonam Drive	Santa F	Argentina
KIMBERLY LEE	KIMBERLY.LEE@sakilacustomer.org	934730187245	96 Tafuna Way	Crdoba	Argentina
LEONARD SCHOFIELD	LEONARD.SCHOFIELD@sakilacustomer.org	779461480495	88 Nagaon Manor	Tandil	Argentina
JORDAN ARCHULETA	JORDAN.ARCHULETA@sakilacustomer.org	817740355461	1229 Varanasi (Benares) Manor	Avellaneda	Argentina
LYDIA BURKE	LYDIA.BURKE@sakilacustomer.org	686015532180	1483 Pathankot Street	San Miguel de Tucumn	Argentina
WILLIE MARKHAM	WILLIE.MARKHAM@sakilacustomer.org	296394569728	1623 Kingstown Drive	Almirante Brown	Argentina
FLORENCE WOODS	FLORENCE.WOODS@sakilacustomer.org	330838016880	1532 Dzerzinsk Way	Merlo	Argentina
WILLIE HOWELL	WILLIE.HOWELL@sakilacustomer.org	991802825778	1244 Allappuzha (Alleppey) Place	Vicente Lpez	Argentina
STEPHANIE MITCHELL	STEPHANIE.MITCHELL@sakilacustomer.org	42384721397	42 Brindisi Place	Yerevan	Armenia
JILL HAWKINS	JILL.HAWKINS@sakilacustomer.org	931059836497	1440 Compton Place	Linz	Austria
AUDREY RAY	AUDREY.RAY@sakilacustomer.org	493008546874	1010 Klerksdorp Way	Graz	Austria
NORA HERRERA	NORA.HERRERA@sakilacustomer.org	621625204422	1587 Loja Manor	Salzburg	Austria

Customer	Contact Details	Customer address	Customer city	Customer Country
AARON SELBY	AARON.SELBY@sakilacustomer.org	409315295763	1519 Santiago de los Caballeros Loop	Mwene-Ditu	Congo, The Democratic Republ
ic of the				
ADAM GOOCH	ADAM.GOOCH@sakilacustomer.org	166898395731	230 Urawa Drive	Adoni	India
ADRIAN CLARY	ADRIAN.CLARY@sakilacustomer.org	182059202712	1986 Sivas Place	Udine	Italy
AGNES BISHOP	AGNES.BISHOP@sakilacustomer.org	778502731092	866 Shivapuri Manor	Sambhal	India
ALAN KAHN	ALAN.KAHN@sakilacustomer.org	464511145118	753 Ilorin Avenue	Emeishan	China
ALBERT CROUSE	ALBERT.CROUSE@sakilacustomer.org	256546485220	1641 Changhwa Place	Bamenda	Cameroon
ALBERTO HENNING	ALBERTO.HENNING@sakilacustomer.org	618156722572	502 Mandi Bahauddin Parkway	Barcelona	Venezuela
ALEX GRESHAM	ALEX.GRESHAM@sakilacustomer.org	118011831565	251 Florencia Drive	Uruapan	Mexico
ALEXANDER FENNELL	ALEXANDER.FENNELL@sakilacustomer.org	575081026569	231 Kaliningrad Place	Bergamo	Italy
ALFRED CASILLAS	ALFRED.CASILLAS@sakilacustomer.org	129673677866	1727 Matamoros Place	Sawhaj	Egypt
ALFREDO MCADAMS	ALFREDO.MCADAMS@sakilacustomer.org	324346485054	1407 Surakarta Manor	Serpuhov	Russian Federation
ALICE STEWART	ALICE.STEWART@sakilacustomer.org	171822533480	1135 Izumisano Parkway	Fontana	United States
ALICIA MILLS	ALICIA.MILLS@sakilacustomer.org	761379480249	1963 Moscow Place	Nagaon	India
ALLAN CORNISH	ALLAN.CORNISH@sakilacustomer.org	50898428626	947 Trshavn Place	Tarlac	Philippines

Q7. Using a subquery list all cities from the United Kingdom along with the city ID.

```sql
SELECT
  ci.city AS "City",
  ci.city_id AS "City ID"
FROM
  city ci
WHERE
  ci.country_id = (
    SELECT
      co.country_id
    FROM
      country co
    WHERE
      co.country = 'United Kingdom'
  );
```

City	City ID
Bradford	88
Dundee	149
London	312
Southampton	494
Southend-on-Sea	495
Southport	496
Stockport	500
York	589

(8 rows)

Q8. As an extension of the previous query, list the cities from the United Kingdom and France using the IN operator, this time showing the country name as well. Group them by country.

```
SELECT
  ci.city_id AS "City ID",
  ci.city AS "City",
  co.country AS "Country"
FROM
  city ci
  JOIN country co ON ci.country_id = co.country_id
WHERE
  co.country IN ('United Kingdom', 'France')
ORDER BY
  co.country,
  ci.city;
```

City ID	City	Country
92	Brest	France
297	Le Mans	France
543	Toulon	France
544	Toulouse	France
88	Bradford	United Kingdom
149	Dundee	United Kingdom
312	London	United Kingdom
494	Southampton	United Kingdom
495	Southend-on-Sea	United Kingdom
496	Southport	United Kingdom
500	Stockport	United Kingdom
589	York	United Kingdom

(12 rows)

Challenge Questions - This is not marked as part of the lab but is taken into account for general marking.

C1 - Using subqueries, list the concatenated first name and last name along with the email address - as a separate column - of all customers that have rented an action movie. List them in alphabetical order first name then last name.

Note: This query is a difficult one and can be solved in multiple ways and multiple subqueries. As long as you are using at least one subquery for the correct output, it will be a valid query. Look for multiple usage of **IN** and **USING** operators.

LAB 7(21/11/2024)
SECURITY

Lab description

Online syntax and technical documentations: https://devdocs.io/

NOTE: *Using `movie_rental` database (LAB 5), you will need to change between superuser role and the new created roles. You can use two terminals. Look for specific syntaxes for login with other roles and the roles permissions in the lecture slides or documentation.*

Q1. Create read-only access to the "staff" table to a new role called "junior_analyst". In order to achieve this you will need to follow a couple of steps. The steps order is given but you will need to write the code (syntaxes). Use 2 terminals - one for super admin (your UP number, connected to the movie_rental database) and one for the created role.

a. -- Create the Role ***junior_analyst*** with a Password and login rights
   - CODE:

b. – Login with junior_analyst role (using 2nd terminal)
   - `psql -h localhost -p 5432 -U junior_analyst -d movie_rental`

c. – Try to select the staff table
   - `SELECT * FROM staff;`
   - You should have the following error: "permission denied for table staff"

d. -- Grant SELECT on the table STAFF to the role ***junior_analyst:***
   - CODE:

e. – Select staff table again. You should be able to see staff table data.

ANSWERS:

```
CREATE ROLE junior_analyst WITH LOGIN PASSWORD '123r';
```

```
CREATE ROLE junior_analyst WITH LOGIN PASSWORD '1234';
```

```
You are now connected to database "movie_rental" as user "up2246775".
movie_rental=# CREATE ROLE junior_analyst  WITH LOGIN PASSWORD '123r';
CREATE ROLE
movie_rental=#
```

```
GRANT CONNECT ON  DATABASE movie_rental TO junior_analyst;
```

```
movie_rental=# GRANT CONNECT ON  DATABASE movie_rental TO junior_analyst;
GRANT
```

```
movie_rental=> SELECT * FROM staff;
ERROR:  permission denied for table staff
movie_rental=>
```

```
movie_rental=# GRANT SELECT ON staff TO junior_analyst;
GRANT
```

```
staff_id | first_name | last_name | address_id |           email            | store_id | active | username |
    password                |        last_update         |     picture
---------+------------+-----------+------------+----------------------------+----------+--------+----------+---
-----------------------------+----------------------------+-------------------
       1 | Mike       | Hillyer   |          3 | Mike.Hillyer@sakilastaff.com |        1 | t      | Mike     | 8c
9ca88db6464eac60da96345513964 | 2022-05-16 15:13:11.79328+00 | \x89504e470d0a5a0a
       2 | Jon        | Stephens  |          4 | Jon.Stephens@sakilastaff.com |        2 | t      | Jon      | 8c
9ca88db6464eac60da96345513964 | 2022-05-16 15:13:11.79328+00 |
(2 rows)
```

Q2. Following a similar process as above, how would you grant **INSERT** access to the "***payment_p2022_01***" table to a new role called "***cashier***"?

Use 2 terminals again

a. Create the role with the correct logins and access level
   - CODE:

b. Grant the insert access for ***`cashier`*** role for the table ***`payment_p2022_01`***.
   - CODE:

c. Login with junior_analyst role (using 2nd terminal)
   - `psql -h localhost -p 5432 -U cashier -d movie_rental`

d. Try to make the following INSERT. Depending on how you granted the permissions, you might have an error. Fix the error on grant. If no error is generated and the INSERT was made, move to question 3.

```
INSERT INTO payment_p2022_01(customer_id, staff_id, rental_id, amount,
payment_date) VALUES (1, 1, 1, 5.99, CURRENT_TIMESTAMP);
```

```
ERROR:  permission denied for table payment_p2022_01
movie_rental=>
```

Q3. Using your cashier role, try to `SELECT * FROM rental;`. You should not be able to do so.

a. Switch to your superuser role and allow the role cashier to make SELECT and INSERT from the rental table.

- CODE:

```
GRANT INSERT ON rental TO cashier;
GRANT SELECT ON rental TO cashier;
```

b. Using the 2nd terminal (with ***cashier*** role) make the following INSERT:

```
INSERT INTO rental (rental_id, rental_date, inventory_id, customer_id,
return_date,     staff_id, last_update)
VALUES (17000, CURRENT_DATE,  1,  1, NULL, 1, CURRENT_TIMESTAMP);

- see if the insert was made (you should have the today date)
SELECT * FROM rental WHERE rental_id = 17000;
```

c. Using the role of cashier, DELETE the record with ID 17000

```
DELETE FROM rental WHERE rental_id = 17000;
```

d. It should fail. How you can fix this and delete the record using cashier role?

Q4. How would you give a role named "***manager***" the ability to update the "***country***" table and also give them permission to grant this privilege to another role called "***sales***"?

a. Create the role ***manager*** with all necessary permissions (login, update and to grant permissions to others)

- CODE:`CREATE ROLE manager WITH LOGIN PASSWORD '1234';`

```
GRANT CONNECT ON  DATABASE movie_rental TO manager  ;
GRANT UPDATE ON country to manager WITH GRANT OPTION;
```

b. Create a role ***sales*** but only login permissions

- CODE:

```
CREATE ROLE sales  WITH LOGIN PASSWORD '1234';
GRANT CONNECT ON  DATABASE movie_rental TO sales  ;
```

c. Login as **manager** and grant the UPDATE on COUNTRY permission for ***sales*** role

```
psql -h localhost -p 5432 -U manager -d movie_rental
```

- CODE: `GRANT UPDATE ON country to sales;`
-

d. Login as ***sales*** make the following update—

```
psql -h localhost -p 5432 -U sales -d movie_rental
UPDATE country SET last_update = NOW() WHERE country_id = 102;
```

Most likely you will have an error - Why? Think to the Transaction Analysis. Adjust the privileges for 'manager' and 'sales' and perform the UPDATE from 'sales' account.

e. Check if the update was successful as **sales** - for last_update you should have the current data and time.

```
SELECT * FROM country WHERE country_id=102;
```

```
 country_id |    country     |          last_update          | country_code
------------+----------------+-------------------------------+--------------
        102 | United Kingdom | 2024-12-11 15:16:37.234162+00 | UK
(1 row)
```

Q5. Using the superuser role, create the following VIEW:

```
CREATE VIEW customer_view AS SELECT * FROM customer;
```

a. How you will allow the "sales" role the ability to access the view without granting them direct access to the table?

- CODE: `GRANT SELECT ON customer_view to manager WITH GRANT OPTION;`
- `GRANT SELECT ON customer_view TO sales;`
-

b. `SELECT * FROM customer;` should fail with "*ERROR: permission denied for table customer*"

`SELECT * FROM customer_view;` Should work without any issue

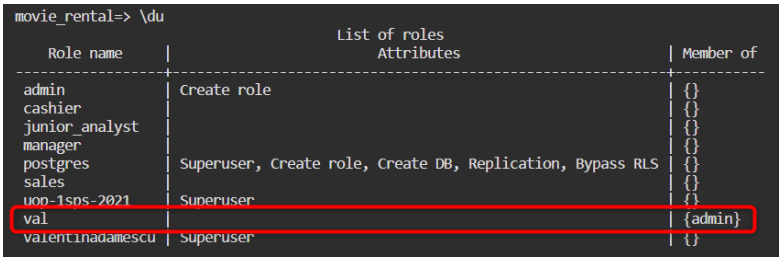
Q6. As superuser add a new staff member

```
INSERT INTO staff (staff_id, first_name, last_name, address_id, email, store_id,
active, username, password, last_update, picture)
VALUES (3, 'Val','Adam', 5,'val.adam@sakilastaff.com', 1,'t', 'valadam',
'8cb2237d0679ca88db6464eac60da96345513964', CURRENT_TIMESTAMP, NULL);
```

a. Create a new role as "admin" that will have the capability to create new roles.

b. Assign the staff named "Val" to admin group.

c. If everything was executed correctly, running the **\du** command you should have the following entry



```
movie_rental=> \du
                                   List of roles
    Role name     |                         Attributes                         | Member of
------------------+------------------------------------------------------------+-----------
 admin            | Create role                                                | {}
 cashier          |                                                            | {}
 junior_analyst   |                                                            | {}
 manager          |                                                            | {}
 postgres         | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
 sales            |                                                            | {}
 uop-1sps-2021    | Superuser                                                  | {}
 val              |                                                            | {admin}
 valentinadamescu | Superuser                                                  | {}
```

NOTE: *In PostgreSQL* ***the users*** *and* ***the groups*** *have both name as ROLE - I know it doesn't make sense, but this is the official description "PostgreSQL manages database access permissions using the concept of roles. A role can be thought of as either a database user, or a group of database users, depending on how the role is set up Source: PostgreSQL"*

Q7. How would you create a role "senior_analyst" that is able to create new tables, drop the created tables, view data in those tables but without the ability to read or modify the data from existing tables?

a. After role creation try to `CREATE TABLE new_table ();` It should work

b. `SELECT * FROM customer;` it should fail

LAB 8 (3/10/2024)

LAB 8 - Data Manipulation and Query Output Formatting

For this week we will use the movie_rental database. Make sure that you have the database installed and you are connected to it. You can use this ERD and this interactive Data Dictionary.

Q1. List **id, first name and last name** of all actors that have the first name as **Scarlett**. The query should ignore the name capitalisation. Hint: *Look for the **ILIKE/LIKE** keywords or wildcard characters.*

ANSWER :

```
SELECT
a.actor_id as "ID",
a.first_name as "First name",
a.last_name as "Last name"
from actor a
WHERE a.first_name ILIKE  'Scarlett';
```

```
 ID  | First name | Last name
-----+------------+-----------
  81 | SCARLETT   | DAMON
 124 | SCARLETT   | BENING
(2 rows)
```

Q2. How many ***unique last name***s are in actors' names? (e.g. If they are 3 Smith's will be counted only once).

ANSWER:

```
SELECT
  count (DISTINCT a.last_name) as "Unique Last name "
from
  actor a;

/* Unique Last name
------------------
               121
(1 row)*/
```

Q3. Following from the previous query, we know how many unique last names are in the database. However, how many of them are appearing only once? E.g. OLIVIER will be twice, so is not that unique, but PESCI will be only once making truly unique. List all the true unique names)

List in alphabetical order the last names that are repeated only once (e.g. if a last name is more than once in the database it will not be considered).

ANSWER:

```
SELECT
    a.last_name AS "Unique Last Names"
FROM
    actor a
GROUP BY
    a.last_name
HAVING
    COUNT(a.last_name) = 1
ORDER BY
    a.last_name;

--(66 rows)
```

Q4. List the **first name and last name** of the actor that appear in most movies and the **number of movies**. Hint: Look for ***USING***.

ANSWER :

```
SELECT
a.first_name as "First name",
a.last_name as "Last name",
COUNT(ma.movie_id) as "number of movie"
from actor a
join movie_actor ma
USING(actor_id)
group by a.last_name, a.first_name
ORDER BY "number of movie" DESC
LIMIT 1;
```

```
 First name | Last name | number of movie
------------+-----------+-----------------
 SUSAN      | DAVIS     |              54
(1 row)
```

Q5. Each store has several copies of each movie. List the available **store 1 inventory** for the movie named '**Purple Movie**'. How many copies are available and what are the inventory IDs?

Answer :

```sql
SELECT
  i.inventory_id,
  COUNT(i.inventory_id) AS "Number of copies"
FROM
  inventory i
  JOIN movie m ON i.movie_id = m.movie_id
  JOIN store s ON i.store_id = s.store_id
WHERE
  m.title = 'Purple Movie'
  AND s.store_id = 1
GROUP BY
  i.inventory_id;

/*inventory_id | Number of copies
--------------+------------------
(0 rows)*/
```

Q6. List staff name and last name, along with their home address, city and email address. The data should be presented in a nice format (exactly as below).

Answer :

```sql
SELECT
  CONCAT (s.first_name, ' ', s.last_name) AS "Staff Name",
  CONCAT(a.address, ' ', a.address2) AS "Staff Address",
  c.city AS "City",
  s.email AS "Staff Email"
FROM
  staff s
  join address a on s.address_id = a.address_id
  join city c on c.city_id = a.city_id;
```

Staff Name	Staff Adress	Staff City	Staff Email
Mike Hillyer	23 Workhaven Lane	Lethbridge	mike.hillyer@sakilastaff.com
Jon Stephens	1411 Lillydale Drive	Woodridge	jon.stephens@sakilastaff.com

Q7. List the ***name and the last name*** in alphabetical order (on surname) of all actors that have acted in the movie named '**Agent Truman**'. The names should appear in a single column just the name of the actors, not the movie.

Challenge Questions - This is not marked as part of the lab but is taken into account for general marking.

C1. - Question 7 displays a column of all actors in a movie, which is not well formatted.
List the movie “Agent Truman” and all the actors as one row. *The output should be exactly as below*.
Hint: look for arrays and strings display.

```
+-------------+-----------------------------------------------------------------------------------------------------+
| Movie Title | Actors in the movie                                                                                 |
|-------------+-----------------------------------------------------------------------------------------------------|
| Agent Truman | Kenneth Hoffman, Sandra Kilmer, Jayne Neeson, Warren Nolte, Kirsten Paltrow, Reese West, Morgan Williams |
+-------------+-----------------------------------------------------------------------------------------------------+
SELECT 1
```

```sql
SELECT
    m.title AS "Movie Title",
    STRING_AGG(CONCAT(a.first_name, ' ', a.last_name), ', ') AS "Actors in the movie"
FROM
    movie m
JOIN
    movie_actor ma ON m.movie_id = ma.movie_id
JOIN
    actor a ON ma.actor_id = a.actor_id
WHERE
    m.title = 'AGENT TRUMAN'
GROUP BY
    m.title;
```

Online documentation: https://devdocs.io/

LAB 9 (5/12/2024)- Advanced SQL

For this week we will use the movie_rental database. Make sure that you have the database installed and you are connected to it. You can use this ERD and Data Dictionary.

Q1. List all the functions available in the movie_rental database.

```
/*movie_rental=# \df
                                                                List of functions
 Schema |            Name            | Result data type |                        Argument data types                         | Type
--------+----------------------------+------------------+--------------------------------------------------------------------+------
 public | get_customer_balance       | numeric          | p_customer_id integer, p_effective_date timestamp with time zone   | func
 public | group_concat               | text             | text                                                               | agg
 public | group_concat               | text             | text, text                                                         | func
 public | inventory_held_by_customer | integer          | p_inventory_id integer                                             | func
 public | inventory_in_stock         | boolean          | p_inventory_id integer                                             | func
 public | last_day                   | date             | timestamp with time zone                                           | func
 public | last_updated               | trigger          |                                                                    | func
 public | movie_in_stock             | SETOF integer    | p_movie_id integer, p_store_id integer, OUT p_movie_count integer  | func
 public | movie_not_in_stock         | SETOF integer    | p_movie_id integer, p_store_id integer, OUT p_movie_count integer  | func
 public | rewards_report             | SETOF customer   | min_monthly_purchases integer, min_dollar_amount_purchased numeric | func
(10 rows)*/
```

Q2. In the database is a function (***movie_in_stock***), with two INT parameters (***p_movie_id*** and ***p_store_id***), that will return all available copies of a given movie in a particular store. Using the named function, find the number of copies of the movie Named "***Angels Life***" in store 1.
Hint: Look for named notation syntax format to call a function parameter.
The output should look like below.

```
select movie_in_stock(25,1);
```

```
 movie_in_stock
----------------
            124
            125
            126
            127
(4 rows)
```

```
+------------------+
| Movies in Stock  |
|------------------|
| 124              |
| 125              |
| 126              |
| 127              |
+------------------+
SELECT 4
```

Q3. Create a stored procedure (sp_add_new_actor) that will automatically insert a new actor.Insert your own name through stored procedure **CALL** e.g. `CALL sp_add_new_actor('Val', 'Adamescu');`
You don't need to allocate any ID as is automatically added by SERIAL PK and the last_update is automatically updated by another function. The procedure should take only two parameters (First Name & Last Name).

ANSWER:

```
CREATE OR REPLACE procedure sp_add_new_actor (
    first_name  VARCHAR,
    last_name VARCHAR
)
Language plpgsql
AS $$
BEGIN
    INSERT INTO actor (first_name, last_name)
    VALUES (first_name, last_name);
END;
$$;

CALL sp_add_new_actor('Reni', 'Todorova');
```

```
 actor_id | first_name | last_name |          last_update
----------+------------+-----------+-------------------------------
      201 | Reni       | Todorova  | 2024-12-07 15:37:15.275148+00
(1 row)
```

Q4. Create a new column in the country table named country_code as VARCHAR.

```
ALTER TABLE
  country
ADD
  country_code VARCHAR;
```

Q5. oops, my bad.. CHAR(2) data type will be more efficient. Modify the country_code from VARCHAR to CHAR(2) and to have only unique values for the column country_code. Use **\d country** and take a screenshot of the table details.

```
    Column    |          Type          | Collation | Nullable |                 Default
--------------+------------------------+-----------+----------+-----------------------------------------
 country_id   | integer                |           | not null | nextval('country_country_id_seq'::regclass)
 country      | character varying(100) |           | not null |
 last_update  | timestamp with time zone |         | not null | now()
 country_code | character(2)           |           |          |
Indexes:
    "country_pkey" PRIMARY KEY, btree (country_id)
Referenced by:
    TABLE "city" CONSTRAINT "city_country_id_fkey" FOREIGN KEY (country_id) REFERENCES country(country_id) ON UPDATE CASCADE ON DELETE RESTRICT
Triggers:
    last_updated BEFORE UPDATE ON country FOR EACH ROW EXECUTE FUNCTION last_updated()
```

Q6. Now if everything is ready, insert the country code as **UK** for the United Kingdom and create the output as country id, country name and country code.

```
 country_id |    country     |          last_update          | country_code
------------+----------------+-------------------------------+--------------
        102 | United Kingdom | 2024-12-07 16:03:28.638401+00 | UK
(1 row)
```

Q7. We are planning some migration of our data but we don't want to transfer everything. Create a new table (new_staff) and copy only id, first & last name and email address.

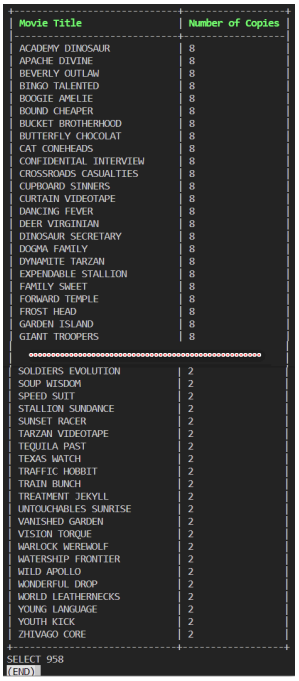
```
CREATE TABLE
  new_staff AS
SELECT
  staff_id,
  first_name,
  last_name,
  email
FROM
  staff;
```

Q8. We have so many copies of each movie for rental but how many? The output of the movies in inventory must be exactly as below. Column naming, text formatting, number of copies starting from the movies with most copies until the ones with less copies.

```
SELECT
 m.title AS "Movie title",
COUNT(i.inventory_id) AS "Number of copies"
FROM movie m
JOIN inventory i ON i.movie_id = m.movie_id
GROUP BY m.title
ORDER BY COUNT(i.inventory_id) DESC, m.title;
```

Movie title	Number of copies
ACADEMY DINOSAUR	8
APACHE DIVINE	8
BEVERLY OUTLAW	8
BINGO TALENTED	8
BOOGIE AMELIE	8
BOUND CHEAPER	8
BUCKET BROTHERHOOD	8
BUTTERFLY CHOCOLAT	8
CAT CONEHEADS	8
CONFIDENTIAL INTERVIEW	8
CROSSROADS CASUALTIES	8
CUPBOARD SINNERS	8
CURTAIN VIDEOTAPE	8
DANCING FEVER	8
DEER VIRGINIAN	8
DINOSAUR SECRETARY	8
DOGMA FAMILY	8

:



Movie Title	Number of Copies
ACADEMY DINOSAUR	8
APACHE DIVINE	8
BEVERLY OUTLAW	8
BINGO TALENTED	8
BOOGIE AMELIE	8
BOUND CHEAPER	8
BUCKET BROTHERHOOD	8
BUTTERFLY CHOCOLAT	8
CAT CONEHEADS	8
CONFIDENTIAL INTERVIEW	8
CROSSROADS CASUALTIES	8
CUPBOARD SINNERS	8
CURTAIN VIDEOTAPE	8
DANCING FEVER	8
DEER VIRGINIAN	8
DINOSAUR SECRETARY	8
DOGMA FAMILY	8
DYNAMITE TARZAN	8
EXPENDABLE STALLION	8
FAMILY SWEET	8
FORWARD TEMPLE	8
FROST HEAD	8
GARDEN ISLAND	8
GIANT TROOPERS	8
SOLDIERS EVOLUTION	2
SOUP WISDOM	2
SPEED SUIT	2
STALLION SUNDANCE	2
SUNSET RACER	2
TARZAN VIDEOTAPE	2
TEQUILA PAST	2
TEXAS WATCH	2
TRAFFIC HOBBIT	2
TRAIN BUNCH	2
TREATMENT JEKYLL	2
UNTOUCHABLES SUNRISE	2
VANISHED GARDEN	2
VISION TORQUE	2
WARLOCK WEREWOLF	2
WATERSHIP FRONTIER	2
WILD APOLLO	2
WONDERFUL DROP	2
WORLD LEATHERNECKS	2
YOUNG LANGUAGE	2
YOUTH KICK	2
ZHIVAGO CORE	2

SELECT 958
(END)

Q9. What is the average movie length per category? Round it to the nearest two decimal places in descending order as below.

```
SELECT
```

```
  c.name AS "Category",
  ROUND(AVG(m.length), 2) AS "Avarage Movie Lenght in Minutes"
FROM
  movie m
  JOIN movie_category mc on mc.movie_id = m.movie_id
  JOIN category c on mc.category_id = c.category_id
GROUP BY
  c.name;
```

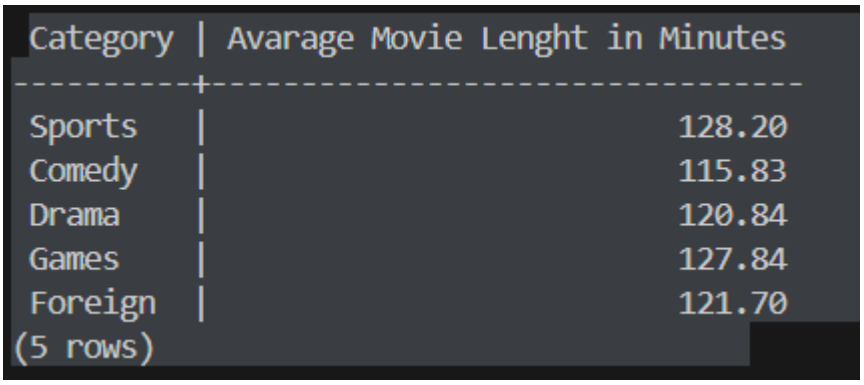
Category	Avarage Movie Lenght in Minutes
Sports	128.20
Classics	111.67
New	111.13
Family	114.78
Comedy	115.83
Animation	111.02
Travel	113.32
Music	113.65
Drama	120.84
Horror	112.48
Sci-Fi	108.20
Games	127.84
Documentary	108.75
Foreign	121.70
Action	111.61
Children	109.80

(16 rows)

Category	Average movie length in Minutes
Sports	128.20
Games	127.84
Foreign	121.70
Drama	120.84
Comedy	115.83
Family	114.78
Music	113.65
Travel	113.32
Horror	112.48
Classics	111.67
Action	111.61
New	111.13
Animation	111.02
Children	109.80
Documentary	108.75
Sci-Fi	108.20

SELECT 16

Q10. We know that the average length of all movies is 115.27. Which categories have movies above average? Do not use LIMIT but select only categories that are above the average in descending order and rounded to nearest two decimal places.
Hint: This query can be achieved in multiple ways. You can use HAVING, USING, and subqueries.



```
 Category | Avarage Movie Lenght in Minutes
----------+---------------------------------
 Sports   |                          128.20
 Comedy   |                          115.83
 Drama    |                          120.84
 Games    |                          127.84
 Foreign  |                          121.70
(5 rows)
```