De la creación a la manipulación de una base de datos relacional

NOMBRE Y APELLIDOS: GLORIA MARIA MANRESA SANTAMARIA

EJERCICIO 1 (25%)

En primer lugar, se ha creado la base de datos (CREATE DATABASE pec2) en un script. Posteriormente se ha ejecutado el código descrito a continuación.

Por último, se ha ejecutado el script proporcionado “BBDD_Erp_data.sql”.

```
-- PEC 2. Ejercicio 1.

-- CREATE DATABASE pec2

-- erp. Crear esquema

CREATE SCHEMA erp;
SET search_path TO erp;

-- tb_cars

CREATE TABLE tb_cars(
            cars_registration CHARACTER(7),
            cars_model CHARACTER VARYING(50) NOT NULL,
            cars_function CHARACTER(20) NOT NULL,
            cars_deposit INTEGER NOT NULL,
            cars_fuel CHARACTER(10) NOT NULL DEFAULT 'gasoil',
            cars_date_input DATE NOT NULL DEFAULT CURRENT_DATE,
            cars_employee CHARACTER VARYING(50),
            cars_date_registration DATE NOT NULL,
            CONSTRAINT pk_tb_cars PRIMARY KEY(cars_registration)
);
```

```
-- tb_gas_station

CREATE TABLE tb_gas_station(
            gas_id CHARACTER(5),
            gas_name CHARACTER VARYING(50) NOT NULL,
            CONSTRAINT pk_tb_gas_station PRIMARY KEY(gas_id),
            CONSTRAINT u_gas_name UNIQUE(gas_name)

);

-- tb_pump

CREATE TABLE tb_pump(
            pm_id INTEGER,
            pm_parent_id INTEGER,
            gas_id CHARACTER(5) NOT NULL,
            pm_descr CHARACTER VARYING(20) NOT NULL,
            CONSTRAINT pk_tb_pump PRIMARY KEY (pm_id),
            CONSTRAINT fk_tb_pump FOREIGN KEY (pm_parent_id) REFERENCES tb_pump(pm_id),
            CONSTRAINT fk_tb_gas_station FOREIGN KEY (gas_id) REFERENCES tb_gas_station(gas_id),
            CONSTRAINT u_gas_id_pm_id UNIQUE (gas_id,pm_id)
);


-- tb_fuel_price

CREATE TABLE tb_fuel_price(
            fp_date DATE,
            fp_fuel CHARACTER(10),
            fp_import NUMERIC(12,3) NOT NULL,
            CONSTRAINT pk_tb_fuel_price PRIMARY KEY (fp_date,fp_fuel)

);

-- tb_refueling

CREATE TABLE tb_refueling(
            gas_id CHARACTER(5) NOT NULL,
            cars_registration CHARACTER(7) NOT NULL,
            pm_id INTEGER NOT NULL,
            rf_liters INTEGER NOT NULL,
            rf_date DATE NOT NULL DEFAULT CURRENT_DATE,
            rf_km INTEGER,
            CONSTRAINT fk_tb_gas_station FOREIGN KEY (gas_id) REFERENCES tb_gas_station(gas_id),
            CONSTRAINT fk_tb_cars FOREIGN KEY (cars_registration) REFERENCES tb_cars(cars_registration),
            CONSTRAINT fk_tb_pump FOREIGN KEY (pm_id) REFERENCES tb_pump(pm_id)
);
-- tb_invoice

CREATE TABLE tb_invoice(
            inv_id INTEGER,
            inv_num CHARACTER(5) NOT NULL,
            inv_date_start DATE NOT NULL,
            inv_date_end DATE NOT NULL,
            inv_amount NUMERIC(12,2) NOT NULL,
            inv_liters_total INTEGER NOT NULL,
            CONSTRAINT pk_tb_invoice PRIMARY KEY (inv_id)
);
```

```
-- tb_lines_invoice

CREATE TABLE tb_lines_invoice(
            inv_id INTEGER,
            linv_id INTEGER,
            cars_registration CHARACTER(7) NOT NULL,
            gas_id CHARACTER(5) NOT NULL,
            linv_liters INTEGER NOT NULL,
            linv_amount NUMERIC(12,2) NOT NULL,
            CONSTRAINT pk_tb_lines_invoices PRIMARY KEY (inv_id,linv_id),
            CONSTRAINT fk_tb_invoice FOREIGN KEY (inv_id) REFERENCES tb_invoice(inv_id),
            CONSTRAINT fk_tb_cars FOREIGN KEY (cars_registration) REFERENCES tb_cars(cars_registration),
            CONSTRAINT fk_tb_gas_station FOREIGN KEY (gas_id) REFERENCES tb_gas_station(gas_id)
)
```

Se ha creado cada tabla según las indicaciones del enunciado. Se han especificado también las restricciones de cada una de ellas, no se ha especificado en las claves primarias que el valor debe ser nulo ya que eso viene dado al tratarse de la clave primaria.

Según comentado anteriormente después de ejecutar este script se ha ejecutado también el proporcionado junto con el enunciado, “BBDD_Erp_data.sql”, para rellenar las estructuras creadas anteriormente con los datos facilitados.

EJERCICIO 2 (25%)

EJERCICIO 2-A

Según solicitado se seleccionan las columnas que incluyen “la matrícula, modelo, empleado que conduce y fecha de matriculación”. Con la cláusula WHERE nos aseguramos que solo aparecen los vehículos de reparto y posteriormente se ordenan por fecha de matriculación descendente.

```
1  -- EJERCICIO 2-A
2  SELECT  cars_registration,
3          cars_model,
4          cars_employee,
5          cars_date_registration
6  FROM erp.tb_cars
7  WHERE cars_function = 'reparto'
8  ORDER BY cars_date_registration DESC
```

Data Output | Messages | Notifications

	cars_registration [PK] character (7)	cars_model character varying (50)	cars_employee character varying (50)	cars_date_registration date
1	3685HDP	MERCEDES-VITO	Antonio García	2011-05-04
2	0815GYR	MERCEDES-VITO	José Pérez	2010-09-11
3	0019GVM	MERCEDES-VITO	Rafael Muñoz	2010-04-23
4	2093GSW	CITROEN-JUMPY	Juan Martínez	2010-01-08
5	9421GMT	MERCEDES-VITO	Jesus Álvarez	2009-06-03
6	4273GFK	MERCEDES-CITAN	David Ruiz	2008-06-20

EJERCICIO 2-B

Para esta consulta necesitamos trabajar sobre 3 tablas, tb_refueling, tb_cars y tb_gas_station.

La tabla rb_refueling tiene las claves foráneas que referencian a las otras dos y como las columnas tienen el mismo nombre podemos realizar un “natural join”.

```
1  -- EJERCICIO 2-B
2  SELECT  c.cars_employee,
3          c.cars_model,
4          g.gas_name,
5          r.rf_date
6  FROM erp.tb_refueling r NATURAL JOIN erp.tb_cars c NATURAL JOIN erp.tb_gas_station g
7  WHERE rf_liters = 41 AND rf_date BETWEEN '2022-08-01' AND '2022-08-31'
8  ORDER BY r.rf_date DESC, c.cars_employee
9
```

Data Output | Messages | Notifications

	cars_employee character varying (50)	cars_model character varying (50)	gas_name character varying (50)	rf_date date
1	Pablo Díaz	BMW-X3	Gasolinera Polígono	2022-08-29
2	Juan Martínez	CITROEN-JUMPY	Gasolinera Estación tren	2022-08-21
3	Jesus Álvarez	MERCEDES-VITO	Gasolinera Norte	2022-08-17
4	Jesus Álvarez	MERCEDES-VITO	Gasolinera Centro	2022-08-08
5	Jesus Álvarez	MERCEDES-VITO	Gasolinera Norte	2022-08-02

EJERCICIO 2-C

Mediante GROUP BY agrupamos las filas según la matrícula (cars_registration) y en la cláusula SELECT se suman los litros repostados por cada coche (matrícula).

También se especifica en la cláusula WHERE que únicamente queremos hacer la consulta en los coches comerciales.

```
1  -- EJERCICIO 2-C
2  SELECT  c.cars_registration,
3          c.cars_employee,
4          SUM(r.rf_liters) AS total_liters
5  FROM erp.tb_refueling r NATURAL JOIN erp.tb_cars c
6  WHERE c.cars_function = 'comercial'
7  GROUP BY c.cars_registration
8  ORDER BY total_liters
9
```

Data Output | Messages | Notifications

	cars_registration [PK] character (7)	cars_employee character varying (50)	total_liters bigint
1	7045KDM	Manuel Rodríguez	174
2	6392KPT	Carmen Ruiz	224
3	8806KZN	Ana Maria Sánchez	315

EJERCICIO 2-D

En este caso hay que hacer la consulta sobre tres tablas, tb_fuel_price, tb_refueling y tb_cars. Debemos combinar los datos de las tablas dos veces puesto que nos piden comparar los vehículos de reparto que en el modelo contenga el texto VITO con todos los vehículos de reparto.

Al comparar dichas tablas se ha especificado en la cláusula WHERE las columnas por las cuales hay que unirlas ya que en este caso los nombres de las columnas no coinciden. También se ha especificado el periodo especificado en el enunciado.

Con la función AVG se ha calculado la media del importe repostado y con la cláusula HAVING se han comparado dichos resultados.

El resultado nos muestra que existen dos vehículos de reparto que en el modelo contienen el texto VITO y que su media del importe repostado en el periodo de septiembre de 2022 es inferior a la media del importe repostado en ese mismo periodo de todos los vehículos de reparto.

```sql
-- EJERCICIO 2-D
SELECT  c1.cars_registration,
        AVG(f1.fp_import*r1.rf_liters) AS avg_import_vito
FROM    erp.tb_fuel_price f1,
        erp.tb_refueling r1,
        erp.tb_cars c1,
        erp.tb_fuel_price f2,
        erp.tb_refueling r2,
        erp.tb_cars c2
WHERE   r1.rf_date = f1.fp_date
        AND r1.cars_registration = c1.cars_registration
        AND c1.cars_fuel = f1.fp_fuel
        AND c1.cars_model LIKE ('%VITO%')
        AND r1.rf_date BETWEEN '2022-09-01' AND '2022-09-30'
        AND r2.rf_date = f2.fp_date
        AND r2.cars_registration = c2.cars_registration
        AND c2.cars_fuel = f2.fp_fuel
        AND r2.rf_date BETWEEN '2022-09-01' AND '2022-09-30'
        AND c1.cars_function ='reparto'
        AND c2.cars_function = 'reparto'
GROUP BY c1.cars_registration
HAVING AVG(f1.fp_import*r1.rf_liters) < AVG(f2.fp_import*r2.rf_liters)
```

Data Output | Messages | Notifications

	cars_registration [PK] character (7)	avg_import_vito numeric
1	0815GYR	69.4050000000000000
2	3685HDP	67.4385000000000000

EJERCICIO 2-E

Se buscan los valores nulos de la columna rf_km para las gasolineras mencionadas en el enunciado.

```
1  -- EJERCICIO 2-E
2  SELECT  cars_registration,
3          gas_id,
4          rf_date
5  FROM    erp.tb_refueling
6  WHERE   rf_km IS NULL
7          AND gas_id IN ('GS04','GS05','GS02')
8  ORDER BY    cars_registration DESC,
9              rf_date
10
```

Data Output | Messages | Notifications

	cars_registration character (7)	gas_id character (5)	rf_date date
1	9421GMT	GS04	2022-09-01
2	8806KZN	GS02	2022-08-26
3	6392KPT	GS05	2022-08-10
4	4273GFK	GS02	2022-08-09
5	0019GVM	GS02	2022-08-06

EJERCICIO 3 (25%)

EJERCICIO 3-A

```
-- EJERCICIO 3A
-- Insertar 3 nuevos vehiculos a erp.tb_cars
INSERT INTO erp.tb_cars
            (cars_registration, cars_model, cars_function,cars_deposit,cars_fuel,
             cars_date_input,cars_employee,cars_date_registration)
    VALUES ('2233JMN','AUDI Q5','gerencia',40,'gasolina',
            to_date('13-11-2022','DD-MM-YYYY'),'Carmen Sevilla Calvo',to_date('16-03-2016','DD-MM-YYYY'));


INSERT INTO erp.tb_cars
            (cars_registration, cars_model, cars_function,cars_deposit,cars_fuel,
             cars_date_input,cars_employee,cars_date_registration)
    VALUES ('7542LSN','AUDI A1','gerencia',30,DEFAULT,
            to_date('13-11-2022','DD-MM-YYYY'),'Carlos Díaz Sevilla',to_date('01-08-2021','DD-MM-YYYY'));


INSERT INTO erp.tb_cars
            (cars_registration, cars_model, cars_function,cars_deposit,cars_fuel,
             cars_date_input,cars_employee,cars_date_registration)
    VALUES ('1974LBN','AUDI A3','gerencia',35,'gasoil',
            to_date('13-11-2022','DD-MM-YYYY'),'Javier Díaz Sevilla',to_date('30-09-2019','DD-MM-YYYY'))
```

Comprobamos que se han insertado correctamente:

```
SELECT *
FROM erp.tb_cars
```

Data Output | Messages | Notifications

	cars_registration [PK] character (7)	cars_model character varying (50)	cars_function character (20)	cars_deposit integer	cars_fuel character (10)	cars_date_input date	cars_employee character varying (50)	cars_date_registration date
1	3685HDP	MERCEDES-VITO	reparto	60	gasoil	2022-07-22	Antonio García	2011-05-04
2	7045KDM	SKODA-FABIA	comercial	50	gasoil	2022-07-23	Manuel Rodríguez	2017-10-05
3	2093GSW	CITROEN-JUMPY	reparto	60	gasoil	2022-07-22	Juan Martínez	2010-01-08
4	8806KZN	SEAT-IBIZA	comercial	30	gasolina	2022-07-24	Ana Maria Sánchez	2019-08-12
5	0815GYR	MERCEDES-VITO	reparto	40	gasoil	2022-07-25	José Pérez	2010-09-11
6	5649JSN	BMW-X3	gerencia	60	gasolina	2022-07-26	Pablo Díaz	2016-09-08
7	0019GVM	MERCEDES-VITO	reparto	60	gasoil	2022-07-24	Rafael Muñoz	2010-04-23
8	4273GFK	MERCEDES-CITAN	reparto	60	gasoil	2022-07-24	David Ruiz	2008-06-20
9	6392KPT	SKODA-FABIA	comercial	30	gasoil	2022-07-25	Carmen Ruiz	2018-10-03
10	9421GMT	MERCEDES-VITO	reparto	60	gasoil	2022-07-26	Jesus Álvarez	2009-06-03
11	2233JMN	AUDI Q5	gerencia	40	gasolina	2022-11-13	Carmen Sevilla Calvo	2016-03-16
12	7542LSN	AUDI A1	gerencia	30	gasoil	2022-11-13	Carlos Díaz Sevilla	2021-08-01
13	1974LBN	AUDI A3	gerencia	35	gasoil	2022-11-13	Javier Díaz Sevilla	2019-09-30

EJERCICIO 3-B

Se borran de la tabla tb_fuel_price las filas en las que no se cumpla que f.fp_date = r.rf_date. De manera que si en una determinada fecha f.fp_date no se realizó ningún repostaje y, por lo tanto, no existe ningún r.rf_date que corresponda con la misma fecha, se borrará dicha entrada.

```
1  -- EJERCICIO 3B
2
3  DELETE FROM erp.tb_fuel_price f
4  WHERE NOT EXISTS (  SELECT r.rf_date
5                      FROM erp.tb_refueling r
6                      WHERE f.fp_date = r.rf_date
7                                   )
8
```

Comprobamos que se han borrado las filas correctamente:

```
1  SELECT DISTINCT f.fp_date
2  FROM            erp.tb_fuel_price f
3  WHERE NOT EXISTS (  SELECT r.rf_date
4                      FROM erp.tb_refueling r
5                      WHERE f.fp_date = r.rf_date
6                                   )
```

Data Output | Messages | Notifications

fp_date
date

EJERCICIO 3-C

```sql
-- EJERCICIO 3-C

-- Añadir columna fp_import_without_vat
ALTER TABLE erp.tb_fuel_price
ADD COLUMN fp_import_without_vat NUMERIC(12,3);

-- Rellenamos los datos de la columna
UPDATE erp.tb_fuel_price
SET fp_import_without_vat = (fp_import)/1.21;

-- Añadir restricción, no acepta nulos
ALTER TABLE erp.tb_fuel_price
ALTER COLUMN fp_import_without_vat SET NOT NULL

```

Comprobamos que la columna se ha añadido correctamente.

```sql
SELECT *
from erp.tb_fuel_price

```

Data Output | Messages | Notifications

	fp_date [PK] date	fp_fuel [PK] character (10)	fp_import numeric (12,3)	fp_import_without_vat numeric (12,3)
1	2022-08-01	gasoil	1.929	1.594
2	2022-08-02	gasoil	1.949	1.611
3	2022-08-03	gasoil	1.969	1.627
4	2022-08-04	gasoil	1.989	1.644
5	2022-08-05	gasoil	2.009	1.660
6	2022-08-06	gasoil	2.029	1.677
7	2022-08-07	gasoil	1.999	1.652
8	2022-08-08	gasoil	1.969	1.627
9	2022-08-09	gasoil	1.939	1.602
10	2022-08-10	gasoil	1.909	1.578

EJERCICIO 3-D

Creamos una restricción en la tabla tb_cars de manera que únicamente se acepten o bien cars_function = 'gerencia' o bien que el campo cars_employee no sea nulo. De manera que este campo puede ser únicamente nulo en caso de que se cumpla cars_function = 'gerencia'.

```
-- EJERCICIO 3-D
ALTER TABLE erp.tb_cars
ADD CONSTRAINT cars_employee_null CHECK
(cars_function = 'gerencia' OR cars_employee IS NOT NULL)

```

Comprobamos la restricción creada anteriormente intentando añadir una fila siendo cars_function = 'comercial' y cars_employee NULL. Nos devuelve un error según lo esperado:

```
INSERT INTO erp.tb_cars
            (cars_registration, cars_model, cars_function,cars_deposit,cars_fuel,
             cars_date_input,cars_employee,cars_date_registration)
    VALUES ('2233JMN','AUDI Q5','comercial',40,'gasolina',
            to_date('13-11-2022','DD-MM-YYYY'),NULL,to_date('16-03-2016','DD-MM-YYYY'));
```

Data Output | Messages | Notifications

```
ERROR:  el nuevo registro para la relación «tb_cars» viola la restricción «check» «cars_employee_null»
DETAIL:  La fila que falla contiene (2233JMN, AUDI Q5, comercial           , 40, gasolina  , 2022-11-13, null, 2016-03-16).
SQL state: 23514
```

EJERCICIO 3-E

Se ha creado el usuario solicitado dándole acceso únicamente a las tablas y los permisos mencionados en el enunciado.

No se ha especificado "WITH GRANT OPTION" para asegurarnos de que no puede asignar permisos a otros usuarios.

```
-- EJERCICIO 3-E
CREATE USER registerer WITH PASSWORD '1234';
GRANT SELECT ON erp.tb_refueling TO registerer;
GRANT INSERT ON erp.tb_refueling TO registerer;
GRANT UPDATE ON erp.tb_refueling TO registerer;
GRANT SELECT ON erp.tb_cars TO registerer;

```

EJERCICIO 4 (25%)

EJERCICIO 4-1

Cualquier combinación puede ser interna o externa, independientemente de si se trata de una combinación natural o no.

Inner join o combinación interna.

Una combinación interna devuelve únicamente las filas que cumplen la condición especificada.

Por ejemplo, si comparamos las tablas Work_team_a y Work_team_b suponiendo que a_id = b_id entonces obtendríamos la siguiente tabla:

Tabla: Inner join (a_id = b_id)			
Campo: a_id	Campo: employee_a	Campo: b_id	Campo: employee_b
1	Andrea Mendoza	1	Claudia Camacho
2	Martin Lara	2	Carlos Alvarado
3	Lucas Rojas	3	Andrea Mendoza
4	Carlos Alvarado	4	Pedro Lombardi

Esto es porque los valores 1,2,3,4 coinciden en ambas tablas. En cambio, si la comparación se hiciese con la condición, employee_a = employee_b quedaría de la siguiente manera:

Tabla: Inner join (employee_a = employee_b)			
Campo: a_id	Campo: employee_a	Campo: b_id	Campo: employee_b
1	Andrea Mendoza	3	Andrea Mendoza
4	Carlos Alvarado	2	Carlos Alvarado

En este caso se han perdido todas las filas en las que no se cumple la condición employee_a = employee_b.

La combinación interna puede ser, o no, natural (NATURAL INNER JOIN). La combinación natural se trata de una equicombinación que consiste en combinar las tablas en función de una columna con el mismo nombre que las relaciona. Este tipo de combinación elimina las columnas repetidas.

Outer join o combinación externa

La combinación externa puede ser de tres tipos, combinación externa izquierda, combinación externa derecha y combinación externa plena.

Según hemos visto, en el "inner join" se pierden las filas de las dos tablas que no cumplan la condición. Si queremos conservar las filas de una tabla determinada habrá que hacer una combinación externa.

Por ejemplo, la combinación externa izquierda conserva todas las filas de la tabla que se sitúa a la izquierda del "LEFT OUTER JOIN" y únicamente devuelve las que cumplan la condición de la derecha. La condición se describe después del "ON":

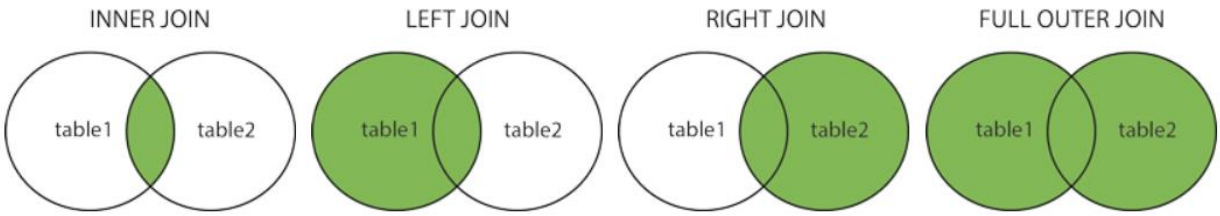
Work_team_a **LEFT OUTER JOIN** Work_team_b **ON** employee_a=employee_b			
Campo: a_id	Campo: employee_a	Campo: b_id	Campo: employee_b
1	Andrea Mendoza	3	Andrea Mendoza
2	Martin Lara	NULL	NULL
3	Lucas Rojas	NULL	NULL
4	Carlos Alvarado	2	Carlos Alvarado

Como podemos observar la tabla de la izquierda (Work_team_a) está completa mientras que la tabla de la derecha (Work_team_b) solo presenta las filas que cumplen la condición employee_a=employee_b.

En el caso de querer conservar todas las filas de ambas tablas se realizaría una combinación externa plena. Por ejemplo:

Work_team_a **FULL OUTER JOIN** Work_team_b **ON** employee_a=employee_b			
Campo: a_id	Campo: employee_a	Campo: b_id	Campo: employee_b
1	Andrea Mendoza	3	Andrea Mendoza
4	Carlos Alvarado	2	Carlos Alvarado
2	Martin Lara	NULL	NULL
3	Lucas Rojas	NULL	NULL
NULL	NULL	1	Claudia Camacho
NULL	NULL	4	Pedro Lombardi

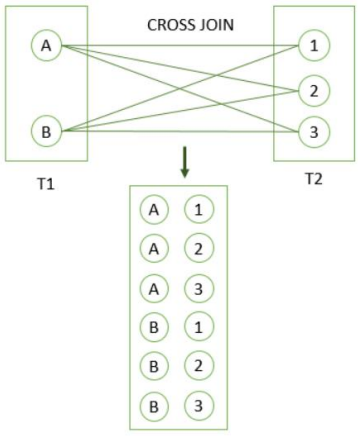
Una manera gráfica y que define muy bien los diferentes tipos de inner y outer join se puede encontrar en el siguiente enlace https://www.w3schools.com/sql/sql_join.asp. Se trata de la siguiente imagen:



CROSS JOIN

Cross join permite realizar el producto cartesiano de las filas de dos o más tablas. En el caso de las tablas work_team_a y work_team_b el producto cartesiano de estas daría como resultado una tabla con 4x4=16 filas.

La siguiente imagen, recogida del siguiente enlace, https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-cross-join/ , ilustra muy bien el funcionamiento de Cross Join:



Cross join devuelve todas las posibles combinaciones que existen entre las filas que pertenecen a las tablas, sin compararlas de ninguna manera. Un cross join entre las tablas work_team_a y work_team_b quedaría de la siguiente manera:

Work_team_a **CROSS JOIN** Work_team_b			
Campo: a_id	Campo: employee_a	Campo: b_id	Campo: employee_b
1	Andrea Mendoza	1	Claudia Camacho
2	Martin Lara	1	Claudia Camacho
3	Lucas Rojas	1	Claudia Camacho
4	Carlos Alvarado	1	Claudia Camacho
1	Andrea Mendoza	2	Carlos Alvarado
2	Martin Lara	2	Carlos Alvarado
3	Lucas Rojas	2	Carlos Alvarado
4	Carlos Alvarado	2	Carlos Alvarado
1	Andrea Mendoza	3	Andrea Mendoza
2	Martin Lara	3	Andrea Mendoza
3	Lucas Rojas	3	Andrea Mendoza
4	Carlos Alvarado	3	Andrea Mendoza
1	Andrea Mendoza	4	Pedro Lombardi
2	Martin Lara	4	Pedro Lombardi
3	Lucas Rojas	4	Pedro Lombardi
4	Carlos Alvarado	4	Pedro Lombardi

EJERCICIO 4-2

Las opciones LIMIT y OFFSET permiten devolver únicamente una selección de filas de la consulta. Por ejemplo, si al final de una consulta SELECT indicas LIMIT 10 la consulta mostrará únicamente 10 filas. El OFFSET indica el número de filas que se deben "saltar" antes de mostrar las "n" filas que indique el LIMIT.

Recordamos que las filas de una tabla no están ordenadas a menos que en la consulta se ordenen (ORDER BY). Si la intención es devolver un número de filas concretas será necesario combinar las opciones LIMIT/OFFSET con ORDER BY.

Según la tabla Work_team_a, si quisiéramos ver únicamente las dos primeras filas suponiendo que están ordenadas de manera ascendente por a_id se debería especificar al final de la consulta ORDER BY a_id LIMIT 2 y devolvería lo siguiente:

Campo: a_id	Campo: employee_a
1	Andrea Mendoza
2	Martin Lara

En caso de escribir ORDER BY a_id LIMIT 2 OFFSET 2 devolvería lo siguiente:

Campo: a_id	Campo: employee_a
3	Lucas Rojas
4	Carlos Alvarado

Criterios de valoración

En el enunciado se indica el peso/valoración de cada ejercicio.

Para conseguir la puntuación máxima en los ejercicios, es necesario explicar con claridad la solución que se propone.

Formato y fecha de entrega

Tenéis que enviar la PEC al buzón de Entrega y registro de EC disponible en el aula (apartado Evaluación). El formato del archivo que contiene vuestra solución puede ser **.pdf, .doc y .docx**. **Para otras opciones, por favor, contactar previamente con vuestro consultor**.

La fecha límite para entregar la PEC2 es el **21/11/2022**.

Nota: **Propiedad intelectual**

Al presentar una práctica o PEC que haga uso de recursos ajenos, se tiene que presentar junto con ella un documento en que se detallen todos ellos, especificando el nombre de cada recurso, su autor, el lugar donde se obtuvo y su estatus legal: si la obra está protegida por el copyright o se acoge a alguna otra licencia de uso (Creative Commons, licencia GNU, GPL etc.). El estudiante tendrá que asegurarse que la licencia que sea no impide específicamente su uso en el marco de la práctica o PEC. En caso de no encontrar la información correspondiente tendrá que asumir que la obra está protegida por el copyright.

Será necesario, además, adjuntar los ficheros originales cuando las obras utilizadas sean digitales, y su código fuente, si así corresponde.