



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

NOMBRE Y APELLIDOS: Jorge Oliva Ramos

EJERCICIO 1 (20%)

El resultado de este ejercicio se adjunta en el fichero : pec2_ej1.sql

EJERCICIO 2 (30%)

a) Proporcionar el identificador, el episodio, el estado y la fecha de creación de las peticiones de prestaciones, ordenadas ascendentemente, desde el 1 de enero de 2019 hasta la fecha de ejecución de la consulta.

pgAdmin interface showing a SQL query execution. The query is:

```

1 /* EJERCICIO 2a Entendemo que solo nos interesa los registros de las peticiones (status = "Solicitada ") */
2 SELECT order_id identificador, encounter_id episodio, status estado ,created_dt fecha
3 FROM clinical.tb_orders
4 WHERE status= 'Solicitada' and
5        created_dt between '2019-01-01 23:59:59' and NOW()
6 ORDER BY created_dt ASC
7
8

```

The output shows columns: identificador (integer), episodio (integer), estado (character varying (50)), and fecha (timestamp without time zone). A messages panel on the right indicates successful execution with 0 rows affected.



b) Proporcionar la lista de los pacientes a los que se les pidió una prueba de categoría Laboratorio o de M.Interna con anterioridad a 2015. y que finalmente fue cancelada. Concretamente queremos que aparezca: el nombre del paciente, la categoría de la prueba, la descripción, el identificador de la prestación, el tipo de episodio, el estado de la prestación y la fecha en que se canceló.

The screenshot shows the pgAdmin 4 interface with a SQL query executed in the Query Editor. The query filters for canceled tests of categories 'Laboratorio' or 'M.Interna' created before 2015. The results are displayed in a table with 7 columns: paciente, categoria, descripcion, identificador, episodio, tipo, and estado.

| paciente | categoria | descripcion | identificador | episodio | tipo | estado |
|----------|-------------|-----------------------|---------------|----------|------------------|-----------|
| Claudia | M.Interna | Consulta sucesiva MIN | 357 | 78919 | Urgencia | Cancelada |
| Ada | Laboratorio | Hemograma_San | 361 | 952783 | Consulta Externa | Cancelada |
| Ada | Laboratorio | Creatinina_Srm | 363 | 952783 | Consulta Externa | Cancelada |

c) Proporcionar la lista de los tres facultativos que más han gastado en pruebas. Queremos su nombre y especialidad, junto al número de pruebas que ha pedido, el coste de todas ellas y el número de pacientes diferentes a que corresponden. Ordenados de mayor a menor por el coste de las pruebas.

The screenshot shows the pgAdmin 4 interface with a SQL query executed in the Query Editor. The query ranks doctors by the total cost of tests they ordered, showing the top 3. The results are displayed in a table with 6 columns: doctor, especialidad, pruebas, importe, pacientes, and a placeholder for a 6th column.

| doctor | especialidad | pruebas | importe | pacientes | |
|--------------|-----------------------------|---------|-------------------|-----------|--|
| Dra. Rivera | Radiodiagnóstico | 66 | 9208.975891113281 | 18 | |
| Dra. Adriana | Cirugía General y Digestiva | 53 | 5814.099029541016 | 15 | |
| Dra. Barceló | Cirugía General y Digestiva | 47 | 5173.76008605957 | 13 | |



d) Proporcionar la lista con todos los datos de los pacientes diferentes de la aseguradora Mapfre que no han sido atendidos nunca de manera urgente.

The screenshot shows the PgAdmin interface with a SQL query executed in the Query Editor. The query selects patient information from the 'clinical' database, excluding those insured by 'Mapfre' who have not been attended in an urgent manner. The results are displayed in the Data Output pane.

```

1 /*EJERCICIO 2d */
2 SELECT DISTINCT p.patient_id paciente, p.ehr_number ehr, p."name" nombre, p.sex sexo,
3 p.birth_dt fecha_nacimiento, p.residence direccion, p.insurance aseguradora
4 FROM
5 clinical.tb_patient p, clinical.tb_encounter e
6 WHERE p.insurance='Mapfre' and
7 e.encounter_type <> 'Urgencia'

```

| | paciente integer | ehr integer | nombre character varying (50) | sexo character (1) | fecha_nacimiento date | direccion character varying (100) | aseguradora character varying (50) |
|---|------------------|-------------|-------------------------------|--------------------|-----------------------|-----------------------------------|------------------------------------|
| 1 | 19 | 1018 | Antonia | F | 1998-11-07 | Carrer Principal | Mapfre |
| 2 | 2 | 1002 | Carlos | M | 1989-02-10 | Carrer de dalt | Mapfre |
| 3 | 31 | 1030 | Josep | M | 1949-11-07 | Carrer Principal | Mapfre |
| 4 | 14 | 1013 | Sheila | F | 1951-11-07 | Carrer de dalt | Mapfre |
| 5 | 8 | 1007 | Sara | F | 1965-11-07 | Carrer de dalt | Mapfre |

Messages: Successfully run. Total query runtime: 114 msec. 5 rows affected.

e) Proporcionar el número de historial clínico, el nombre, la edad de los pacientes y el número de veces que han sido atendidos en la clínica, de aquellos a los que se ha atendido más de 8 veces, ordenados por el número de veces que han sido atendidos

The screenshot shows the PgAdmin interface with a SQL query executed in the Query Editor. The query selects patient information from the 'clinical' database, ordered by the number of times they have been attended in descending order, where the number of attendances is greater than 8. The results are displayed in the Data Output pane.

```

1 /*EJERCICIO 2e No se especifica en el ejercicio pero he ordenado descendientemente por
2 de esta consulta es ver las personal que mas atenciones tienen */
3 SELECT p.ehr_number historial, p."name" nombre, date_part('year',age(p.birth_dt)) edad,
4 COUNT(e.patient_id) atenciones
5 FROM
6 clinical.tb_patient p, clinical.tb_encounter e
7 WHERE p.patient_id = e.patient_id
8 GROUP BY p.ehr_number, p."name", p.birth_dt
9 HAVING COUNT(e.patient_id) > 8
10 ORDER BY atenciones DESC

```

| | historial integer | nombre character varying (50) | edad double precision | atenciones bigint |
|---|-------------------|-------------------------------|-----------------------|-------------------|
| 1 | 1015 | Ada | | 46 |
| 2 | 1005 | Marta | | 11 |
| 3 | 1014 | Antonio | | 10 |
| 4 | 1003 | Eva | | 9 |

Messages: Successfully run. Total query runtime: 121 msec. 4 rows affected.



EJERCICIO 3 (30%)

a) Se va a ofrecer un nuevo servicio en la clínica, para ello deberemos añadir una nueva especialidad (*Digestología*) de la cual pasará a encargarse el Dr. Peris, y se ofrecerá una nueva prueba diagnóstica (*Colonoscopia*). Así pues se deben insertar la nueva especialidad y la nueva prueba según la especificación adjunta, y se debe modificar la especialidad del Dr. Peris (con `user_id = 13`) que ahora dirigirá esta nueva especialidad de Digestología. En total deberéis hacer tres sentencias.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, including the 'test_clinical' schema. The central 'Query Editor' contains the following SQL query:

```

1 /*EJERCICIO 3a) */
2 select s.med_service_id ,u.full_name , c.order_desc from clinical.tb_medical_services s,clinical.tb_orders_catalog c
3 clinical.tb_orders_catalog c
4 WHERE u.user_id = 13 and
5 u.med_service_id = s.med_service_id and
6 c.order_code=9001;
7
8

```

The 'Messages' pane on the right indicates: 'Successfully run. Total query runtime: 99 msec. 1 rows affected.'

The 'Data Output' pane at the bottom shows the result of the query:

| | med_service_id | full_name | order_desc |
|---|----------------|------------------------|------------------------|
| 1 | integer | character varying (50) | character varying (50) |
| | 6 | Dr. Peris | Colonoscopia |



b) Nos comentan que se debe de **alterar**, en una **única sentencia**, la tabla `tb_encounter` para controlar que `discharge_dt` pueda ser nulo, pero nunca sea inferior a `arrival_dt`.

← → ↻ ↗ 127.0.0.1:61432/browser/

PGAdmin File Object Tools Help

Browser

- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Procedures
- Sequences
- Tables (6)
 - tb_encounter**
 - tb_medical_services
 - tb_orders
 - tb_orders_catalog
 - tb_patient
 - tb_users
- Trigger Functions
- Types
- Views
- public
- test_clinical
- videogames

tb_encounter

General Columns Advanced Constraints Parameters Security SQL

Inherited from table(s) Select to inherit from...

| | Name | Data type | Length/Precision | Scale | Not NULL? | Prim |
|--|----------------|-----------------------------|------------------|-------|-----------|------|
| | encounter_id | integer | | | Yes | |
| | patient_id | integer | | | Yes | |
| | encounter_type | character varying | 50 | | Yes | |
| | arrival_dt | timestamp without time zone | | | Yes | |
| | discharge_dt | timestamp without time zone | | | No | |
| | med_service_id | integer | | | Yes | |

Cancel Reset Save

← → ↻ ↗ 127.0.0.1:61432/browser/

PGAdmin File Object Tools Help

Browser

- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Procedures
- Sequences
- Tables (6)
 - tb_encounter**
 - tb_medical_services
 - tb_orders
 - tb_orders_catalog
 - tb_patient
 - tb_users
- Trigger Functions
- Types
- Views
- public
- test_clinical
- videogames

tb_encounter

General Columns Advanced Constraints Parameters Security SQL

Primary Key Foreign Key Check Unique Exclude

| | Name | Check |
|--|-----------|----------------------------|
| | chec_date | discharge_dt >= arrival_dt |

Cancel Reset Save



c) Implementad, en una **única sentencia**, una vista **clinical.top_six_cost_doctors** sobre la consulta c) del ejercicio 2, pero haciendo que aparezcan 6 en vez de 3 médicos. Aseguraos de que los usuarios no puedan insertar valores que inicialmente no deben mostrarse a través de la vista.

The screenshot shows the pgAdmin interface with the 'Query Editor' tab active. The SQL query being executed is:

```
1 CREATE VIEW clinical.top_six_cost_doctors ( doctor, especialidad, pruebas ,importe ,numero_de_pacientes)
2 AS
3 (select u.full_name doctor, s.description especialidad,COUNT(o.order_id)pruebas ,
4 SUM(c.cost)-COUNT(o.order_id) importe ,
5 COUNT(DISTINCT e.patient_id) pacientes
6 from clinical.tb_users u, clinical.tb_medical_services s, clinical.tb_orders o,
7 clinical.tb_orders_catalog c, clinical.tb_encounter e
8 where u.medical_license_nbr!= 'null' and u.med_service_id=s.med_service_id and u.user_id=o.created_by_user
9 o.order_code = c.order_code and o.encounter_id= e.encounter_id and
10 (c.subcategory='Laboratorio' or c.subcategory='Pruebas')
11 GROUP BY u.full_name,s.description
12 ORDER BY importe DESC
13 LIMIT 6);
14
15
```

The 'Messages' pane on the right shows: 'CREATE VIEW Query returned successfully in 118 msec.'

Nota : Esta ultimo requerimiento : “Aseguraos de que los usuarios no puedan insertar valores que inicialmente no deben mostrarse a través de la vista” no entiendo el contexto en el que tiene que ser implementado?

d) Nos piden, que se añada, en una **única sentencia**, una columna en la tabla **tb_users** denominada **mail_address**, para almacenar la dirección de correo electrónico del usuario. Ha de poder almacenar hasta 100 caracteres, no se podrá repetir y por defecto será nulo.

The screenshot shows the pgAdmin interface with the 'Query Editor' tab active. The SQL query being executed is:

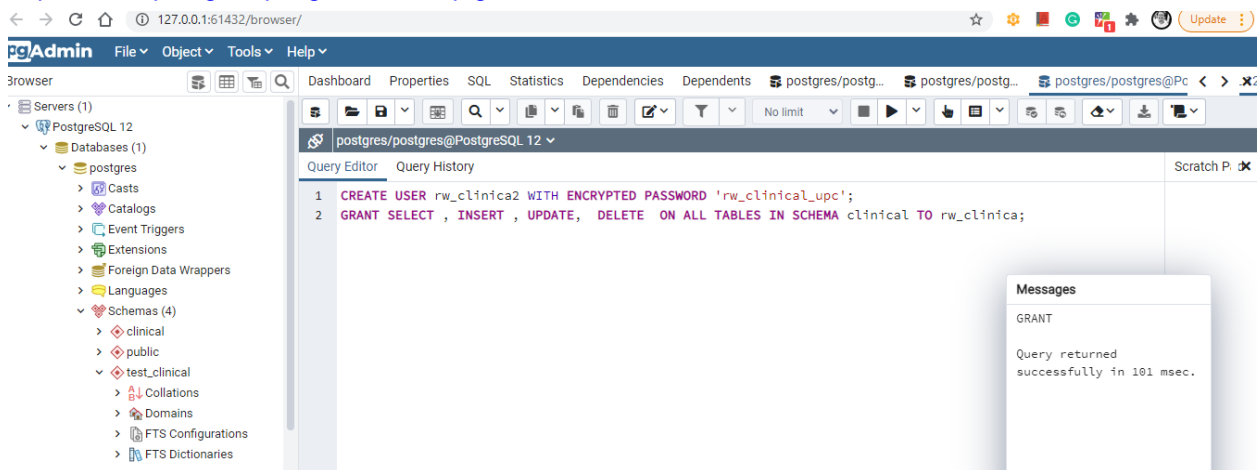
```
1 /*Ejercicioi 3d*/
2 ALTER TABLE clinical.tb_users
3 ADD COLUMN mail_address VARCHAR(100) UNIQUE DEFAULT NULL;
4
```

The 'Messages' pane on the right shows: 'ALTER TABLE Query returned successfully in 115 msec.'



e) Se ha de crear un usuario de sistema (cuidado! no confundir con `tb_users`) ***rw_clinical*** con acceso al esquema ***clinical*** y contraseña ***rw_clinical_uoc***. Este usuario puede realizar lecturas, inserciones, actualizaciones, borrados (nunca truncamientos) de las tablas del esquema ***clinical*** (*rw* significa *read and write*). Asignar los permisos necesarios a dicho usuario mediante SQL. Aseguraos también de que este usuario no pueda asignar permisos a otros usuarios sobre dichas tablas. Referencia a la documentación:

<https://www.postgresql.org/docs/12/sql-grant.html>





EJERCICIO 4 (20%)

Desde la empresa nos piden que realicemos unas tareas de comprensión y práctica sobre PostgreSQL, como nuevos expertos que sois.

Contestad brevemente a los siguientes apartados:

Explicar en qué consisten los predicados: `LIKE`, `ALL`, `ANY/SOME` y `EXISTS` y poned ejemplos de uso sobre la base de datos de la clínica.

LIKE: Esta es una expresión que nos permite ver si una columna cumple alguna característica determinada ósea retornara true si el contenido de la columna hace matching con la condiciones o patrón especificado y retorna false en caso no, Utilizamos expresiones regulares para trabajar con condiciones complejas : En términos que nos interesa False/True significa devuelve el registro o no

Ejemplo: Nos devuelve todos lo registros de los empleado de la clínica cuyo nombre completo empieza con “Dra” (Nos interesa obtener la información de todas las doctoras)

- `SELECT * FROM clinical.tb_users u WHERE u.full_name LIKE 'Dra%'`

ALL: Esta es una expresión que nos permite ver si una columna cumple una condición sobre una subquery, la característica de esta expresión es que todos los registros de la subquery tiene que cumplir la condición para devolver TRUE , en casos contrario devuelve FALSE. En términos que nos interesa TRUE/FALSE devuelve el registro o no.

Ejemplo: Nos devuelve todos lo registros de los paciente que no han tenido ninguna servicio de Urgencia (Nos interesa ofrecer un servicio personalizado a un segmento específico de nuestros clientes)

- `SELECT * FROM clinical.tb_patient p WHERE p.patient_id <> ALL (select e.patient_id from clinical.tb_encounter e where e.encounter_type = 'Urgencia' and e.patient_id=p.patient_id)`



ANY/SOME: Esta es una expresión que nos permite ver si una columna cumple una condición sobre una subquery, la característica de esta expresión es alguno de los registros de la subquery tiene que cumplir la condición para devolver TRUE , en casos contrario devuelve FALSE. En términos que nos interesa TRUE/FALSE devuelve el registro o no.

Ejemplo: Nos devuelve todos los registros de los pacientes que han algún servicio de Urgencia (Nos interesa ofrecer un servicio personalizado a un segmento específico de nuestros clientes)

- `SELECT * FROM clinical.tb_patient p WHERE p.patient_id = ANY (select e.patient_id from clinical.tb_encounter e where e.encounter_type = 'Urgencia' and e.patient_id=p.patient_id)`

EXIST: Es una expresión que nos permite ver si existe algún valor en una subquery , la característica de esta expresión es si alguno de los registros de la subquery existe devuelve TRUE en casos contrario devuelve FALSE.

Ejemplo: Nos devuelve todos los registros de los pacientes que han tenido algún servicio de Urgencia (Nos interesa ofrecer un servicio personalizado a un segmento específico de nuestros clientes)

- `SELECT * FROM clinical.tb_patient p WHERE EXISTS (select e.patient_id from clinical.tb_encounter e where e.encounter_type = 'Urgencia' and e.patient_id=p.patient_id)`

NOTA : Solo comentar que desde mi punto de vista ALL, ANY/SOME / EXIST son bastantes similares, comparan un valor con una lista de valores , podría ser solo aplique sobre una relaciones 1 a N ((diagrama de entidad relación)? ... Estas notas las pongo con la intención de recibir feedback, ya que he considerado importante el feedback de un profesor experto en mi camino de aprendizaje...Gracias!

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

El formato del archivo que contiene vuestra solución puede ser **.pdf, .doc y .docx**. **Para otras opciones, por favor, contactar previamente con vuestro consultor**. El nombre del fichero debe contener el código de la asignatura, vuestro apellido y vuestro nombre, así como el número de actividad (PEC2).

El fichero .zip que contenga todos los ficheros de la PEC (tanto los ficheros .sql como el documento que muestra los resultados de vuestras soluciones) tenéis que enviarlo al buzón de Entrega y registro de EC disponible en el aula (apartado Evaluación).

La fecha límite para entregar la PEC2 es el **11/04/2021**

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.