Nombre: Martínez Martínez Alberto
Materia: Bases de datos
Documento: Serie II

2.- Indicar las ciudades que tienen más de un aeropuerto. Agregar su notación correspondiente en álgebra relacional.

```
SELECT
  current_timestamp AS fecha_hora_sistema,
  current_user AS usuario_actual,
  city,
  COUNT (*) AS numero_aeropuertos
  FROM aeropuertos
  GROUP BY city
  HAVING COUNT (*) >1;
```

	fecha_hora_sistema timestamp with time zone	usuario_actual name	city character varying	numero_aeropuertos bigint
1	2023-12-05 12:28:37.576946-06	postgres	Jackson	2
2	2023-12-05 12:28:37.576946-06	postgres	Springfield	2
3	2023-12-05 12:28:37.576946-06	postgres	Albany	2
4	2023-12-05 12:28:37.576946-06	postgres	Columbia	2
5	2023-12-05 12:28:37.576946-06	postgres	New York	2
6	2023-12-05 12:28:37.576946-06	postgres	Jacksonville	2
7	2023-12-05 12:28:37.576946-06	postgres	Wilmington	2
8	2023-12-05 12:28:37.576946-06	postgres	San Diego	2
9	2023-12-05 12:28:37.576946-06	postgres	Chicago	2
10	2023-12-05 12:28:37.576946-06	postgres	Houston	2
11	2023-12-05 12:28:37.576946-06	postgres	Charleston	2
12	2023-12-05 12:28:37.576946-06	postgres	Portland	2
13	2023-12-05 12:28:37.576946-06	postgres	Columbus	2
14	2023-12-05 12:28:37.576946-06	postgres	Rochester	2

3.- Nombre de las aerolíneas que no terminan en Inc. ni en Co. Agregar su notación correspondiente en álgebra relacional.

```
SELECT
  current_timestamp AS fecha_hora_sistema,
  current_user AS usuario_actual,
  aerolineas.*
 FROM aerolineas
 WHERE NOT (airline LIKE '% Inc.' OR airline LIKE '% Co.');
```

	fecha_hora_sistema timestamp with time zone	usuario_actual name	iata_code character varying	airline character varying
1	2023-12-05 12:31:09.845086-06	postgres	B6	JetBlue Airways
2	2023-12-05 12:31:09.845086-06	postgres	NK	Spirit Air Lines
3	2023-12-05 12:31:09.845086-06	postgres	EV	Atlantic Southeast Airlines
4	2023-12-05 12:31:09.845086-06	postgres	VX	Virgin America

4.- Indicar los nombres de los aeropuertos que estuvieron implicados en el vuelo que presentó el mayor retraso de llegada.

	fecha_hora_del_sistema timestamp with time zone	usuario_actual name	arrivale_delay smallint	origin_airport character varying	destination_airport character varying	nombre_origen character varying	nombre_destino character varying
1	2023-12-05 13:41:07.095335-06	postgres	1971	BHM	DFW	Birmingham-Shuttlesworth International Airport	Dallas/Fort Worth International Airport

```sql
SELECT
    CURRENT_TIMESTAMP AS fecha_hora_del_sistema,
    CURRENT_USER AS usuario_actual,
    arrivale_delay,
    v.origin_airport,
    v.destination_airport,
    origen.airport AS nombre_origen,
    destino.airport AS nombre_destino
FROM vuelos v
JOIN aeropuertos origen ON v.origin_airport = origen.iata_code
JOIN aeropuertos destino ON v.destination_airport = destino.iata_code
WHERE arrivale_delay = (SELECT MAX(arrivale_delay) FROM vuelos);
```

5.- Mostrar aquella categoría (tabla artículo) que tiene el precio mínimo. La información debe estar agrupada (Implica que la consulta no sale con sólo selects y wheres).

Query Query History

```sql
SELECT
  current_timestamp AS fecha_hora_sistema,
  current_user AS usuario_actual,
  categoria,
  MIN(precio) AS precio_mas_bajo
FROM articulo
GROUP BY categoria
ORDER BY precio_mas_bajo
LIMIT 1;
```

	fecha_hora_sistema timestamp with time zone	usuario_actual name	categoria character varying (35)	precio_mas_bajo numeric
1	2023-12-05 12:37:39.725551-06	postgres	accesorios	120

6.- Se desea conocer el nombre de aquellas aerolíneas cuyo segundo carácter del iata code termina en X o 9. Debe incluirse una columna que muestre dicha terminación.

Query Query History

```sql
SELECT
    CURRENT_TIMESTAMP AS fecha_hora_del_sistema,
    CURRENT_USER AS usuario_actual,
    iata_code,
    airline,
    RIGHT(iata_code, 1) AS ultimo_caracter
FROM aerolineas
WHERE (iata_code LIKE '%X' OR iata_code LIKE '%9');
```

	fecha_hora_del_sistema timestamp with time zone	usuario_actual name	iata_code character varying	airline character varying	ultimo_caracter text
1	2023-12-05 12:55:20.365483-06	postgres	F9	Frontier Airlines Inc.	9
2	2023-12-05 12:55:20.365483-06	postgres	VX	Virgin America	X

7.- Proporcionar el nombre de los aeropuertos cuya latitud se encuentre entre 40 y 41, y su longitud sea menor que el promedio de la longitud. Nota: el promedio se toma de aquellas observaciones cuya latitud se encuentre entre 40 y 41.

Query | Query History

```sql
SELECT
    CURRENT_TIMESTAMP AS fecha_hora_del_sistema,
    CURRENT_USER AS usuario_actual,
    airport
FROM aeropuertos
WHERE (latitude BETWEEN 40 AND 41)
AND
longitude< (SELECT AVG(longitude) FROM aeropuertos WHERE latitude BETWEEN 40 AND 41);
```

	fecha_hora_del_sistema timestamp with time zone	usuario_actual name	airport character varying
1	2023-12-05 13:05:23.741213-06	postgres	Arcata Airport
2	2023-12-05 13:05:23.741213-06	postgres	Elko Regional Airport
3	2023-12-05 13:05:23.741213-06	postgres	Central Nebraska Regional Airport
4	2023-12-05 13:05:23.741213-06	postgres	Yampa Valley Airport (Yampa Valley Regional)
5	2023-12-05 13:05:23.741213-06	postgres	Lincoln Airport (Lincoln Municipal)
6	2023-12-05 13:05:23.741213-06	postgres	Redding Municipal Airport
7	2023-12-05 13:05:23.741213-06	postgres	Salt Lake City International Airport
8	2023-12-05 13:05:23.741213-06	postgres	Valdez Airport

8.- ¿Cuántos aviones por aerolínea y día, fueron cancelados saliendo del aeropuerto de Honolulu?

```sql
SELECT
  CURRENT_TIMESTAMP AS fecha_del_sistema,
  CURRENT_USER AS usuario_actual,
  v.airline AS nombre_aerolinea,
  DATE(v.year || '-' || v.month || '-' || v.day) AS fecha,
  COUNT(*) AS cantidad_aviones_cancelados
FROM vuelos v
JOIN aeropuertos a ON v.origin_airport = a.iata_code
WHERE v.cancelled = '1'
  AND a.city = 'Honolulu'
GROUP BY v.airline, fecha
ORDER BY fecha, v.airline;
```

	fecha_del_sistema timestamp with time zone	usuario_actual name	nombre_aerolinea character varying	fecha date	cantidad_aviones_cancelados bigint
1	2023-12-05 21:07:42.663757-06	postgres	UA	2015-01-01	1
2	2023-12-05 21:07:42.663757-06	postgres	UA	2015-01-02	1
3	2023-12-05 21:07:42.663757-06	postgres	UA	2015-01-03	1
4	2023-12-05 21:07:42.663757-06	postgres	HA	2015-01-05	1
5	2023-12-05 21:07:42.663757-06	postgres	US	2015-01-05	1
6	2023-12-05 21:07:42.663757-06	postgres	HA	2015-01-13	2
7	2023-12-05 21:07:42.663757-06	postgres	HA	2015-01-14	1
8	2023-12-05 21:07:42.663757-06	postgres	UA	2015-01-14	1
9	2023-12-05 21:07:42.663757-06	postgres	HA	2015-01-18	1
10	2023-12-05 21:07:42.663757-06	postgres	US	2015-01-18	1
11	2023-12-05 21:07:42.663757-06	postgres	HA	2015-01-21	1
12	2023-12-05 21:07:42.663757-06	postgres	UA	2015-01-24	1
13	2023-12-05 21:07:42.663757-06	postgres	UA	2015-01-26	1
14	2023-12-05 21:07:42.663757-06	postgres	HA	2015-01-27	2
15	2023-12-05 21:07:42.663757-06	postgres	UA	2015-02-01	1
16	2023-12-05 21:07:42.663757-06	postgres	US	2015-02-01	1
17	2023-12-05 21:07:42.663757-06	postgres	HA	2015-02-04	1
18	2023-12-05 21:07:42.663757-06	postgres	UA	2015-02-08	2

Total rows: 145 of 145 | Query complete 00:00:01.003

9.- Hacer un cross join entre la tabla cliente y la tabla aerolíneas. Obviamente ambas tablas forman parte de distintas BDs, debe encontrar la forma de hacerlo.

```sql
-- Instala la extensión dblink si aún no lo has hecho
CREATE EXTENSION IF NOT EXISTS dblink;

-- Establece una conexión a la segunda base de datos (registro_vuelos)
SELECT dblink_connect('host=localhost dbname=registro_vuelos user=postgres password=1234');

-- Realiza el CROSS JOIN entre las tablas de ambas bases de datos
SELECT *
FROM cliente
CROSS JOIN dblink('SELECT * FROM aerolineas'::text) AS t2(iata_code character varying, airline character varying);
```

	id_cliente [PK] character varying (13)	nombre character	ap_pat character	ap_mat character	estado character varying (25)	iata_code character varying	airline character varying
1	aksmvieoci125	Luisa	Balderas	[null]	cdmx	UA	United Air Lines Inc.
2	abcd	mario	martinez	[null]	cdmx	UA	United Air Lines Inc.
3	ejemplo	Jaime	Cruz	flores	nayarit	UA	United Air Lines Inc.
4	aksmvieoci144	Angela	Perez		nayarit	UA	United Air Lines Inc.
5	aksmvieoci126	Luis	Lopez	[null]	cdmx	UA	United Air Lines Inc.
6	aksmvieoci127	Luis	Valderrama	[null]	cdmx	UA	United Air Lines Inc.
7	aksmvieoci137	Luis	Valderrama	[null]	cdmx	UA	United Air Lines Inc.
8	aksmvieoci125	Luisa	Balderas	[null]	cdmx	AA	American Airlines Inc.
9	abcd	mario	martinez	[null]	cdmx	AA	American Airlines Inc.
10	ejemplo	Jaime	Cruz	flores	nayarit	AA	American Airlines Inc.
11	aksmvieoci144	Angela	Perez		nayarit	AA	American Airlines Inc.
12	aksmvieoci126	Luis	Lopez	[null]	cdmx	AA	American Airlines Inc.
13	aksmvieoci127	Luis	Valderrama	[null]	cdmx	AA	American Airlines Inc.
14	aksmvieoci137	Luis	Valderrama	[null]	cdmx	AA	American Airlines Inc.
15	aksmvieoci125	Luisa	Balderas	[null]	cdmx	US	US Airways Inc.
16	abcd	mario	martinez	[null]	cdmx	US	US Airways Inc.
17	ejemplo	Jaime	Cruz	flores	nayarit	US	US Airways Inc.
18	aksmvieoci144	Angela	Perez		nayarit	US	US Airways Inc.

Total rows: 98 of 98 Query complete 00:00:00.279 Ln 13, Col

10.- Cantidad de vuelos cancelados por día

```sql
SELECT
  DATE(year || '-' || LPAD(month::text, 2, '0') || '-' || LPAD(day::text, 2, '0')) AS fecha,
  COUNT(*) AS cantidad_vuelos_cancelados
FROM vuelos
WHERE cancelled = '1'
GROUP BY fecha
ORDER BY fecha;
```

	fecha date	cantidad_vuelos_cancelados bigint
1	2015-01-01	466
2	2015-01-02	257
3	2015-01-03	331
4	2015-01-04	433
5	2015-01-05	435
6	2015-01-06	395
7	2015-01-07	303
8	2015-01-08	813
9	2015-01-09	469
10	2015-01-10	190
11	2015-01-11	354
12	2015-01-12	436
13	2015-01-13	197
14	2015-01-14	178
15	2015-01-15	104
16	2015-01-16	58
17	2015-01-17	32
18	2015-01-18	114

Total rows: 365 of 365 Query complete 00:00:01.119

11.- Seleccionar el nombre de los aeropuertos cuya segunda letra del iata code sea K ´o X, sin usar operadores and, not u or. Puede usar alguna función propia de postgres.

```sql
SELECT
    CURRENT_TIMESTAMP AS fecha_hora_del_sistema,
    CURRENT_USER AS usuario_actual,
    iata_code,
    airport
FROM aeropuertos
WHERE SUBSTRING(iata_code FROM 2 FOR 1) IN ('K', 'X');
```

	fecha_hora_del_sistema timestamp with time zone	usuario_actual name	iata_code character varying	airport character varying
1	2023-12-05 13:11:01.732478-06	postgres	AKN	King Salmon Airport
2	2023-12-05 13:11:01.732478-06	postgres	EKO	Elko Regional Airport
3	2023-12-05 13:11:01.732478-06	postgres	MKE	General Mitchell International Airport
4	2023-12-05 13:11:01.732478-06	postgres	MKG	Muskegon County Airport
5	2023-12-05 13:11:01.732478-06	postgres	OKC	Will Rogers World Airport
6	2023-12-05 13:11:01.732478-06	postgres	RKS	Rock Springs-Sweetwater County Airport
7	2023-12-05 13:11:01.732478-06	postgres	TXK	Texarkana Regional Airport (Webb Field)

12.- Indicar el nombre(s) de la aerolínea cuya distancia de vuelo es la mayor

```sql
SELECT DISTINCT ON (v.airline) v.distance, a.airline,
    CURRENT_TIMESTAMP AS fecha_hora_del_sistema,
    CURRENT_USER AS usuario_actual
FROM vuelos v
JOIN aerolineas a ON v.airline = a.iata_code
WHERE v.distance = (SELECT MAX(distance) FROM vuelos);
```

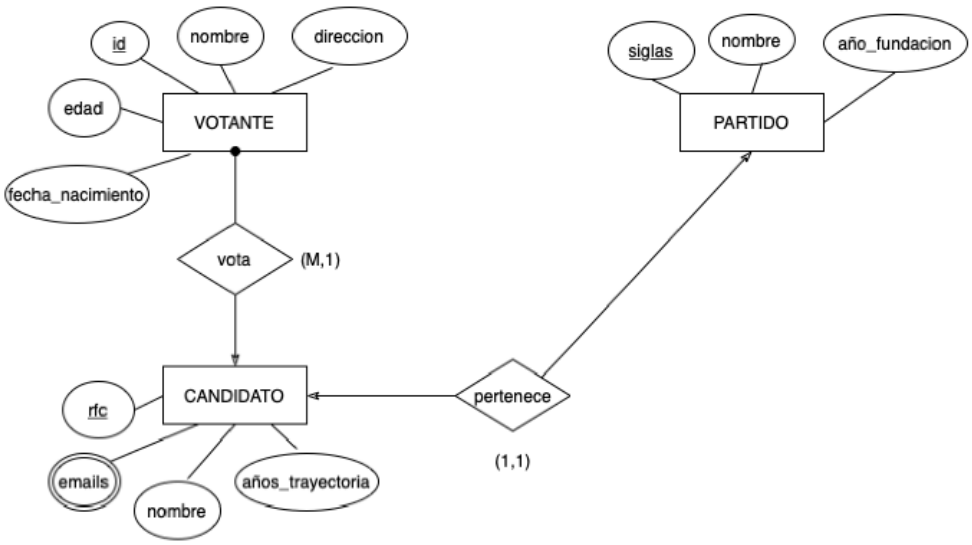
	distance smallint	airline character varying	fecha_hora_del_sistema timestamp with time zone	usuario_actual name
1	4983	Delta Air Lines Inc.	2023-12-05 13:14:04.347782-06	postgres
2	4983	Hawaiian Airlines Inc.	2023-12-05 13:14:04.347782-06	postgres

13.- Indicar el nombre del aeropuerto de origen donde se presentó el mayor tiempo de vuelo.

```sql
SELECT
    CURRENT_TIMESTAMP AS fecha_hora_del_sistema,
    CURRENT_USER AS usuario_actual,
    elapsed_time,
    origin_airport,
    a.airport
FROM vuelos v
JOIN aeropuertos a ON v.origin_airport = a.iata_code
WHERE elapsed_time = (SELECT MAX(elapsed_time) FROM vuelos);
```

	fecha_hora_del_sistema timestamp with time zone	usuario_actual name	elapsed_time smallint	origin_airport character varying	airport character varying
1	2023-12-05 13:26:48.242627-06	postgres	766	EWR	Newark Liberty International Airport

14.- Partiendo del siguiente MER



Modelo Relacional

VOTANTE { id SERIAL [PK],

Nombre VARCHAR (255),

Direccion VARCHAR (255),

Edad smallint,

Fecha:nacimiento DATE,

Candidato VARCAHR (13) [FK]

}

CANDIDATO { rfc VARCHAR (13) [PK]

nombre VARCHAR(255),

años_trayectoria INTEGER

}

emails { id_email SERIAL [PK]

rfc_candidato VARCHAR (13) [FK]

}

PARTIDO { siglas VARCHAR (10) [PK],

Nombre VARCHAR (255),

Anio_fundacion INTEGER,

Candidato VARCHAR(13) [FK]

}

DDL de las relaciones

```
CREATE TABLE votante(
id_votante SERIAL,
nombre VARCHAR(255),
direccion VARCHAR(255),
edad INTEGER,
fecha_nacimiento DATE,
candidato_votante VARCHAR (13),
CONSTRAINT PK_votante PRIMARY KEY (id_votante)
);


CREATE TABLE candidato(
    rfc VARCHAR (13),
    nombre VARCHAR (255),
    anios_trayectoria INTEGER,
    partido_pertenece VARCHAR (10),
    CONSTRAINT PK_candidato PRIMARY KEY (rfc)
);

CREATE TABLE emails(
    id_email SERIAL,
    email VARCHAR(255),
    rfc_candidato VARCHAR(13),
    CONSTRAINT PK_email PRIMARY KEY(id_email)
);

ALTER TABLE emails ADD CONSTRAINT email_fk FOREIGN KEY (rfc_candidato)
REFERENCES candidato (rfc) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;

ALTER TABLE votante ADD CONSTRAINT votante_fk FOREIGN KEY
(candidato_votante)
REFERENCES candidato (rfc) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;

CREATE TABLE partido (
```

```
    siglas VARCHAR (10),
    nombre VARCHAR (255),
    anio_fundacion INTEGER,
    CONSTRAINT PK_partido PRIMARY KEY (siglas)
);

ALTER TABLE candidato ADD CONSTRAINT candidato_fk FOREIGN KEY
(partido_pertenece)
REFERENCES partido (siglas) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
```

15.- Genere una función o procedure que resuelva lo siguiente

```
CREATE OR REPLACE FUNCTION ejercicio15 (categoria_f VARCHAR)
RETURNS VOID
AS
$$
DECLARE
    registro RECORD;
    cur_productos CURSOR FOR SELECT nombre_articulo, precio, categoria FROM
articulo WHERE categoria = categoria_f;
    precio_minimo NUMERIC;
    precio_maximo NUMERIC;
BEGIN

    SELECT MIN(precio), MAX(precio)
    INTO precio_minimo, precio_maximo
    FROM articulo
    WHERE categoria= categoria_f;

    OPEN cur_productos;

    LOOP
        FETCH cur_productos INTO registro;

        EXIT WHEN NOT FOUND;
        RAISE NOTICE 'Nombre: %   Precio: %   Categoria: %, Precio mas bajo:
%,  Precio mas alto: %', registro.nombre_articulo, registro.precio,
registro.categoria, precio_minimo, precio_maximo;

    END LOOP;

    CLOSE cur_productos;
END;
$$
```

```
LANGUAGE plpgsql;
```

```
datos_clase=# SELECT ejercicio15('accesorios');
NOTICE:  Nombre: gorra   Precio: 120   Categoria: accesorios, Precio mas bajo: 120,  Precio mas alto: 600
NOTICE:  Nombre: lentes   Precio: 400   Categoria: accesorios, Precio mas bajo: 120,  Precio mas alto: 600
NOTICE:  Nombre: bolsa   Precio: 300   Categoria: accesorios, Precio mas bajo: 120,  Precio mas alto: 600
NOTICE:  Nombre: mochila   Precio: 570   Categoria: accesorios, Precio mas bajo: 120,  Precio mas alto: 600
NOTICE:  Nombre: Reloj   Precio: 600   Categoria: accesorios, Precio mas bajo: 120,  Precio mas alto: 600
```

16.- Partiendo del siguiente MR, determinar cuál o cuales opciones son correctas para insertar información en las tablas correspondientes.

Primero se deben de colocar los productos dentro de su tabla correspondiente. Antes de intentar hacer alguna inserción a la tabla orden es necesario tener registro de clientes ya que existe la restricción de la llave foránea cliente_fk. Después se agrega la orden y en detalle orden se ponen los productos que conforman la orden del cliente y aquí se debe de tener cuidado que los valores que se presentan en detalle orden coincidan con el precio a pagar en la tabla orden. Para esto se recomienda un trigger que haga esta operación tra un INSERT o UPDATE.