

# Proyecto Bases

---

Archivo `.sql` final: [query.sql](#)

- Miguel Francisco Vargas Contreras
- Nicolas Diaz Granados Cano
- Sara Rodriguez Urueña

## Documentación

Este proyecto fue probado en Oracle 19c Server.

### Creación de tablas

Se crearon dos archivos: `DDL.sql` y `DDL+drop.sql`, ambos destinados a la creación de tablas en la base de datos. La razón para usar dos archivos separados es permitir la limpieza previa de las tablas existentes. Si la base de datos no está vacía, el archivo `DDL+drop.sql` eliminará las tablas existentes antes de crear las nuevas.

El orden utilizado para la creación de las tablas es el siguiente:

1. **Edificio**
2. **Piso**
3. **Cafetería**
4. **Colaborador**
5. **Meta**

Este orden asegura que primero se creen las tablas sin relaciones, facilitando así la creación de las tablas que dependen de ellas. En el archivo `DDL+drop.sql`, las tablas se eliminarán en el orden inverso:

```
DROP TABLE meta;  
DROP TABLE colaborador;  
DROP TABLE cafeteria;  
DROP TABLE piso;  
DROP TABLE edificio;
```

### Edificio

```
CREATE TABLE edificio (  
  id          NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,  
  nombre      VARCHAR2(255) NOT NULL,  
  
  PRIMARY KEY (id)  
);
```

### Piso

```
CREATE TABLE piso (  
  id          NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,  
  numeropiso  NUMBER NOT NULL,  
  idEdificio  NUMBER NOT NULL,  
  
  PRIMARY KEY (id),  
  FOREIGN KEY (idEdificio) REFERENCES edificio (id)  
    ON DELETE SET NULL  
);
```

## Cafetería

```
CREATE TABLE cafeteria (  
  id          NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,  
  nombre      VARCHAR2(255) NOT NULL,  
  idPiso      NUMBER NOT NULL,  
  
  PRIMARY KEY (id),  
  FOREIGN KEY (idPiso) REFERENCES piso (id)  
    ON DELETE SET NULL  
);
```

## Colaborador

```
CREATE TABLE colaborador (  
  id          NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,  
  nombre      VARCHAR2(255) NOT NULL,  
  tipodocumento VARCHAR2(255) NOT NULL,  
  numerodocumento NUMBER NOT NULL,  
  vinculacion  VARCHAR2(255) NOT NULL  
    CHECK (vinculacion IN ('PLANTA', 'TEMPORAL')),  
  comision     NUMBER DEFAULT 10 NOT NULL  
    CHECK (comision >= 0 AND comision <= 100),  
  
  PRIMARY KEY (id)  
);
```

De acuerdo con el enunciado del proyecto, la validación de la vinculación realizará una verificación para asegurar que el dato ingresado sea 'PLANTA' o 'TEMPORAL'.

```
CREATE TABLE meta (  
  id          NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,  
  fechameta   NUMBER NOT NULL,  
  valormeta   NUMBER DEFAULT 0 NOT NULL,  
  valorreal   NUMBER DEFAULT 0 NOT NULL,
```

```
idCafeteria    NUMBER NOT NULL,  
idColaborador  NUMBER NOT NULL,  
  
PRIMARY KEY (id),  
FOREIGN KEY (idCafeteria) REFERENCES cafeteria (id)  
    ON DELETE SET NULL,  
FOREIGN KEY (idColaborador) REFERENCES colaborador (id)  
    ON DELETE SET NULL  
);
```

Usamos **NOT NULL** para evitar la inserción de valores nulos.

## Creación de relaciones

Se utiliza la estructura **INSERT INTO tabla (nombre\_datos...) VALUES (datos...)** porque el **id** se genera automáticamente por la base de datos. Por lo tanto, es necesario especificar los datos y el orden en que se van a insertar.

La inserción de los datos para la tabla **edificio** se realizó manualmente. Posteriormente, se utilizó **ChatGPT** para generar las inserciones para las tablas **piso**, **cafetería**, **meta** y **colaborador**. Esta última tabla requirió un **script** en **TypeScript** para generar fechas aleatorias.

Estas instrucciones se guardan en el archivo **relationsInsertFile.sql**, que comienza eliminando todos los datos de cada tabla.

## Desarrollo de ejercicios

### VISTA\_1

Listado de colaboradores, cafeterías y metas. Liste la cafetería, nombre del edificio, número del piso, nombre del colaborador, número y tipo de documento, fecha de meta, valor de las metas de ventas y valor real de ventas, diferencia porcentual entre meta y valor real. Ordene por fecha de meta, nombre de cafetería nombre de colaborador.

Se uso la definición de variación porcentual tomada de internet: *Se calcula restando el valor antiguo del nuevo y luego, se divide el valor obtenido sobre el valor absoluto antiguo y se multiplica por 100.*

```
((meta.valormeta - meta.valorreal)/meta.valorreal) * 100 as variacionporcentual
```

Ya con esta formula tenemos todos los datos necesarios para la query.

```
SELECT  cafeteria.nombre AS nombreCafeteria,  
        edificio.nombre AS nombreEdificio,  
        piso.numeropiso,  
        colaborador.nombre,  
        colaborador.tipodocumento,  
        colaborador.numerodocumento,  
        meta.fechameta,
```

```

        meta.valormeta,
        meta.valorreal,
        ((meta.valormeta - meta.valorreal)/meta.valorreal) * 100 as
variacionporcentual

FROM cafeteria,
    meta,
    colaborador,
    edificio,
    piso

WHERE cafeteria.id=meta.idCafeteria
    AND colaborador.id=meta.idColaborador
    AND cafeteria.idPiso=piso.id
    AND piso.idEdificio=edificio.id

ORDER BY meta.fechameta,
        cafeteria.nombre,
        colaborador.nombre;
```

[illegible]

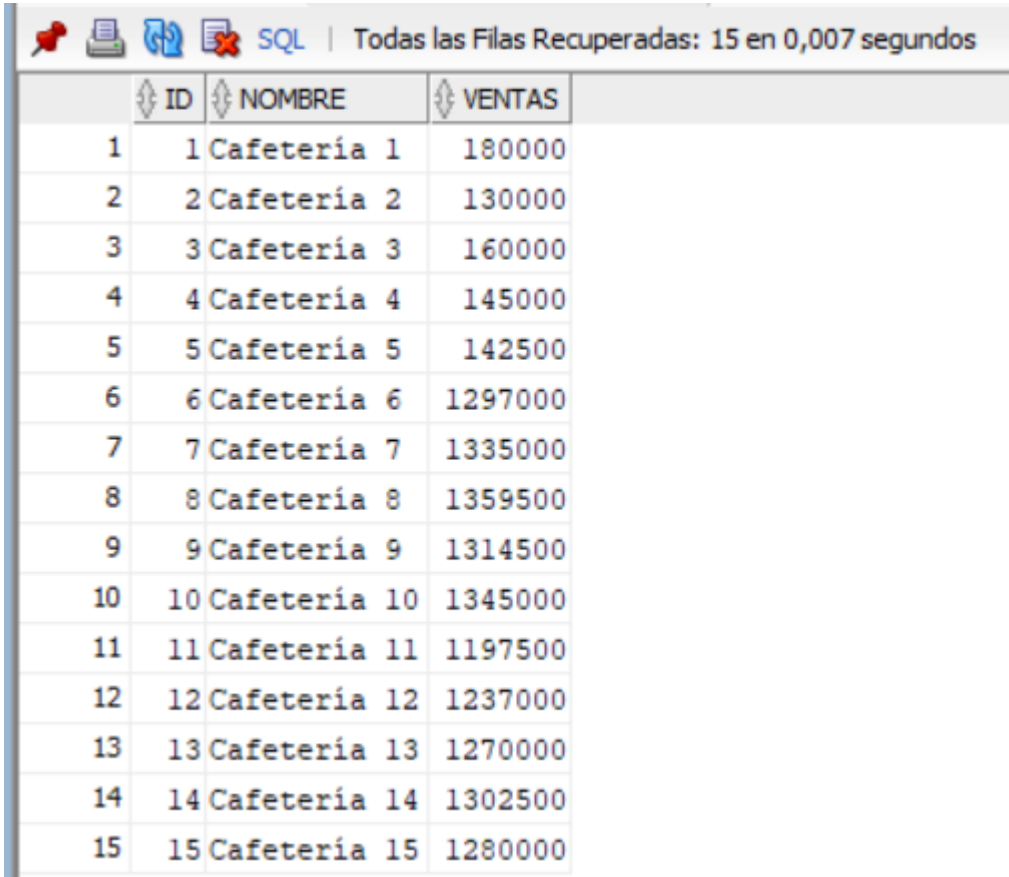
## VISTA 2

Cuales son las ventas totales de cada cafetería?, liste el nombre de la cafetería, total de ventas. Las cafeterías que no tienen ventas deben aparecer en el listado.

Para cambiar el valor `NULL` de las cafeterías sin ventas a `0`, utilizamos la función `COALESCE(SUM(meta.valormeta), 0)`. Esta función calcula la suma de ventas y reemplaza los valores `NULL` por `0`.

Además, usaremos producto cartesiano para combinar los datos de ventas (de la tabla `meta`) con cada cafetería, uniendo las tablas mediante `cafeteria.id = meta.idcafeteria`. Se ordena para una mejor lectura.

```
SELECT cafeteria.id, cafeteria.nombre,  
       COALESCE(SUM(meta.valormeta), 0) AS ventas  
FROM cafeteria, meta  
WHERE cafeteria.id = meta.idcafeteria  
GROUP BY cafeteria.id, cafeteria.nombre  
ORDER BY cafeteria.id;
```



	ID	NOMBRE	VENTAS
1	1	Cafetería 1	180000
2	2	Cafetería 2	130000
3	3	Cafetería 3	160000
4	4	Cafetería 4	145000
5	5	Cafetería 5	142500
6	6	Cafetería 6	1297000
7	7	Cafetería 7	1335000
8	8	Cafetería 8	1359500
9	9	Cafetería 9	1314500
10	10	Cafetería 10	1345000
11	11	Cafetería 11	1197500
12	12	Cafetería 12	1237000
13	13	Cafetería 13	1270000
14	14	Cafetería 14	1302500
15	15	Cafetería 15	1280000

### VISTA\_3

Cuál es el valor por pagar a cada colaborador en cada mes y año? Liste el nombre del colaborador, el valor a pagar (total de ventas multiplicado por la comisión del colaborador). En una última fila muestre el total general de todos los colaboradores.

Conseguimos el valor `anhoMes` eliminando el valor del día en la tupla de la fecha. Para ello, dividimos en 100 y descartamos los decimales usando `ROUND()`. Luego, aplicamos la fórmula para calcular el monto a pagar al colaborador.

Guardamos esta primera consulta con el nombre de `pagos`. Esta tabla contendrá información del colaborador, como el nombre, su comisión, las ventas y el pago a realizar, además de la fecha en formato `añomes`.

Finalmente, uniremos la tabla `pagos` con una nueva query realizando un `SUM(pago)` para encontrar el total de los pagos de todos los colaboradores basandonos en otra query que solo tendrá una fila con el nombre `'TOTAL'`. Mantendremos el valor del pago y cambiamos todos los demás valores por `NULL`, Luego .

```
WITH pagos AS (  
    SELECT nombre,  
           anhoMes,  
           sum(valormeta) AS ventas,  
           comision,  
           ROUND(SUM(valormeta) * ( comision / 100 )) AS pago  
    FROM colaborador, (  
        SELECT idcolaborador,  
               ROUND(fechameta/100) as anhoMes,  
               valormeta
```

```
        FROM meta
    ) m
    WHERE colaborador.id = idcolaborador
    GROUP BY anhoMes, nombre, comision
    ORDER BY nombre, comision
)





SELECT * FROM pagos

UNION ALL

SELECT nombre,
        anho_mes,
        ventas,
        comision,
        SUM(pago) as pago
FROM (
    SELECT 'TOTAL' as nombre,
           null as anho_mes,
           null as ventas,
           null as comision,
           pago from pagos
);
```

Salida de Script x

Resultado de la Consulta x

 SQL | Todas las Filas Recuperadas: 122 en 0,027 segundos

	NOMBRE	ANHOMES	VENTAS	COMISION	PAGO	
102	María Fernández	202203	112000	55	61600	
103	María Fernández	202302	92000	55	50600	
104	María Fernández	202301	317500	55	174625	
105	María Fernández	202202	11000	55	6050	
106	María Fernández	202205	62000	55	34100	
107	María Fernández	202303	102000	55	56100	
108	María Fernández	202106	236000	55	129800	
109	María Fernández	202204	82000	55	45100	
110	María Fernández	202104	12500	55	6875	
111	Roberto García	202105	149000	34	50660	
112	Roberto García	202101	205500	34	69870	
113	Roberto García	202103	19000	34	6460	
114	Roberto García	202204	129000	34	43860	
115	Roberto García	202106	79000	34	26860	
116	Roberto García	202201	89000	34	30260	
117	Roberto García	202304	79500	34	27030	
118	Roberto García	202206	59000	34	20060	
119	Roberto García	202203	109000	34	37060	
120	Roberto García	202102	139000	34	47260	
121	Roberto García	202205	148000	34	50320	
122	TOTAL	(null)	(null)	(null)	5077065	

VISTA\_4





Cuál es el valor de las metas y ventas reales por cada año y mes? Liste año, mes y suma total de las metas, suma total de valores reales y suma total de la diferencia entre el valor real y la meta en ese año – mes.

Seleccionaremos `anhoMes`, `valorreal` y `valormeta` desde una query en la que usaremos nuevamente `ROUND()` para calcular el `añoMes` y haremos la suma de los `valormeta` y `valorreal` este lo agruparemos por `anhoMes` y los organizaremos. Sobre esta tabla, agregaremos una nueva fila que contenga la diferencia entre las sumas de valor real y las sumas de valor meta.

```
SELECT anhoMes,
       sum(valorreal) as ventas,
       sum(valormeta) as metas,
       sum(valormeta) - sum(valorreal) as diferencia
FROM (
    SELECT ROUND(fechameta/100) as anhoMes,
           valorreal,
           valormeta
```

```
FROM meta
) m
GROUP BY anhoMes
ORDER BY anhoMes;
```

Salida de Script x Resultado de la Consulta x

 SQL | Todas las Filas Recuperadas: 23 en 0,004 segundos

	ANHOMES	VENTAS	METAS	DIFERENCIA	
1	202101	393500	396500	3000	
2	202102	477500	480000	2500	
3	202103	698300	703500	5200	
4	202104	582000	588000	6000	
5	202105	504000	508500	4500	
6	202106	682500	687000	4500	
7	202201	525000	540500	15500	
8	202202	726500	751000	24500	
9	202203	943000	958000	15000	
10	202204	1145500	1211000	65500	
11	202205	1079000	1115000	36000	
12	202206	1087500	1094000	6500	
13	202207	139500	150000	10500	
14	202208	100050	120000	19950	
15	202210	120500	183000	62500	
16	202211	168500	170000	1500	
17	202212	140500	150000	9500	
18	202301	786000	791000	5000	
19	202302	541500	544000	2500	
20	202303	531000	535000	4000	
21	202304	1024000	1031500	7500	
22	202305	182500	184000	1500	
23	202306	799500	804000	4500	

VISTA\_5

Cuál es el porcentaje de participación de cada colaborador en el total general? El porcentaje de participación se calcula como la suma total de ventas reales de cada colaborador sobre la suma total de metas en todas las cafeterías. Liste el nombre del colaborador, total de ventas reales y el porcentaje de participación sobre las ventas reales de los colaboradores.

Se calcula la suma total de las ventas, sumando todos los **valorreal** de todas las metas mediante una subconsulta que se asocia como la tabla **m**. Unimos esta subconsulta con las tablas **colaborador** y **meta**, agrupamos por el nombre del colaborador, lo que nos permite conseguir la suma de todas las ventas de cada



colaborador y mantener el total de todas las metas. Calculamos el porcentaje de participación de cada colaborador en las ventas totales dividiendo sus ventas por el total de ventas y multiplicándolo por 100, y mostramos esta información junto con el nombre del colaborador y sus ventas totales.

```
SELECT nombre,
        total,
        sum(valorreal) as ventas,
        (sum(valorreal) * 100 / total) as porcentajeParticipacion
FROM colaborador, meta, (
    SELECT sum(valorreal) as total
    FROM meta
) m
WHERE idColaborador = colaborador.id
GROUP BY nombre, total;
```

 SQL | Todas las Filas Recuperadas: 10 en 0,006 segundos

	NOMBRE	TOTAL	VENTAS	PORCENTAJEPARTICIPACION
1	Jorge Pérez	13377850	1139000	8,51407363664564933827184487791386508295
2	Luis Martínez	13377850	1079500	8,06930859592535422358600223503776765325
3	Fernando Torres	13377850	1168500	8,73458739633050153799003576807932515314
4	Laura Díaz	13377850	1157000	8,64862440526691508725243592954024749866
5	Isabel Morales	13377850	1186500	8,86913816495176728697062681970570756885
6	Carlos López	13377850	1128500	8,43558568828324431803316676446514200712
7	Roberto García	13377850	1197000	8,94762611331417230720930493315443064469
8	Ana Gómez	13377850	3063550	22,90016706720437140497165090055576942483
9	María Fernández	13377850	1111800	8,31075247517351442870117395545622054366
10	Elena Rodríguez	13377850	1146500	8,57013645690451006701375781609152442283

VISTA\_6

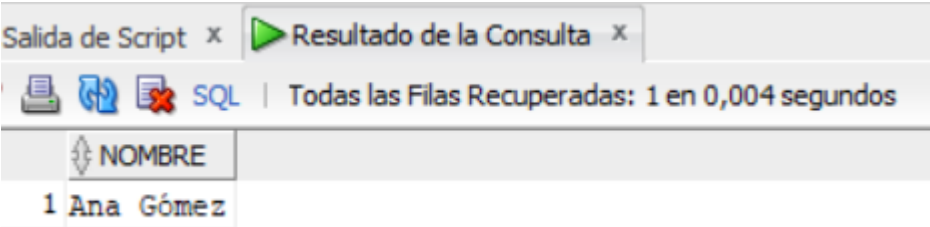
Qué colaborador tiene metas en todas las cafeterías? Liste el nombre del colaborador.

Uniendo las siguientes queries:

- **cantcafeterias:** Contar la cantidad de cafeterias
- **colab:** Producto cartesiano entre colaborador y meta, filtrando para que queden las ventas de cada colaborador, de este solo proyectaremos el nombre, el id del colaborador y el id de la cafeteria, así seleccionando solo los distintos, tendremos una lista de cada cafeteria y cada trabajador que trabajo en ella. Con estas dos queries agruparemos por trabajador y cafeteria, así al contar la cantidad de **idCafeteria** nos dara como resultado la cantidad de cafeterias en las que trabajo cada colaborador y mantendremos **ccafeterias** donde guardamos el total de cafeterias.

Ahora sobre esa query con las cantidades, filtraremos solo los colaboradores que tengan la misma cantidad de cafeterias en las que trabajaron y la cantidad total de cafeterias en la universidad, de esta solo proyectaremos el nombre de los colaboradores que cumplan.

```
SELECT nombre
FROM (
    SELECT DISTINCT nombre,
        count(idCafeteria) as trabajaEn,
        ccafeterias
    FROM (
        SELECT COUNT(cafeteria.id) as ccafeterias
        FROM cafeteria
    ) cantcafeterias, (
        SELECT DISTINCT nombre, colaborador.id, idCafeteria
        FROM colaborador, meta
        WHERE meta.idColaborador=colaborador.id
    ) colab
    GROUP BY nombre, ccafeterias
)
WHERE trabajaen=ccafeterias;
```



VISTA\_7

Edificio	Número total de colaboradores que son de planta	Número total de colaboradores que son temporales	Total
Barón	3	2	5
Giraldo	4	2	6
.....			
Totales	7	4	11

Para calcular la cantidad de colaboradores por tipo de contrato (planta y temporales) y por edificio, crearemos una tabla temporal llamada **colabor**. Esta tabla contendrá todos los colaboradores, el edificio en el que han trabajado, la id del colaborador y su tipo de vinculación.

Una vez creada la tabla **colabor**, necesitaremos calcular una columna de totales. Para ello, utilizaremos dos veces nuestra tabla de cantidades. Esta tabla se llamará **cantidades**.

La tabla **cantidades** se generará utilizando un **Full Outer Join** de dos subconsultas de la tabla **colabor**, filtradas por tipo de contrato. Una subconsulta contendrá la cantidad de colaboradores con vinculación de planta y la otra, la cantidad de colaboradores temporales. Utilizaremos la función **COALESCE** para evitar valores nulos al mostrar los resultados, asegurando así que los edificios sin colaboradores de un tipo específico aparezcan con un valor de 0 en lugar de NULL. El **JOIN** se realizará en función del nombre del edificio.

Finalmente, uniremos la tabla `cantidades` con una tabla agregada que se creará a partir de `cantidades`. Esta nueva tabla reemplazará todos los nombres de los edificios por `'TOTALES'` y seleccionará la suma de las columnas `numeroPlanta`, `numeroTemporal` y `Total`.

```
WITH colabor as (  
    SELECT edificio.id,  
           edificio.nombre,  
           idColaborador,  
           vinculacion  
    FROM meta, colaborador, edificio, piso, cafeteria  
    WHERE colaborador.id = meta.idColaborador AND  
           edificio.id = piso.idEdificio AND  
           piso.id = cafeteria.idPiso AND  
           meta.idcafeteria = cafeteria.id  
)  
cantidades as (  
    SELECT d1.nombre,  
           COALESCE(numeroPlanta, 0) as numeroPlanta,  
           COALESCE(numeroTemporal, 0) as numeroTemporal,  
           COALESCE(numeroPlanta, 0) + COALESCE(numeroTemporal, 0) as Total  
    FROM (  
        SELECT nombre, count(idColaborador) as numeroPlanta  
        FROM colabor  
        WHERE vinculacion = 'PLANTA'  
        GROUP BY nombre  
    ) d1  
    FULL OUTER JOIN (  
        SELECT nombre, count(idColaborador) as numeroTemporal  
        FROM colabor  
        WHERE vinculacion = 'TEMPORAL'  
        GROUP BY nombre  
    ) d2 ON d1.nombre = d2.nombre  
)  
  
SELECT * from cantidades  
UNION (  
    SELECT 'TOTALES' as nombre,  
           sum(numeroPlanta) as numeroPlanta,  
           sum(numeroTemporal) as numeroTemporal,  
           sum(total) as total  
    from cantidades  
)  
);
```

	NOMBRE	NUMEROPLANTA	NUMEROTEMPORAL	TOTAL
1	Ed. Arango Puerta	3	0	3
2	Ed. Atico	1	2	3
3	Ed. Emilio Arangom	17	0	17
4	Ed. Felix Restrepo	1	15	16
5	Ed. Fernando Baron	1	0	1
6	Ed. Gabriel Giraldo	1	0	1
7	Ed. Jorge Hoyoso Vasques	1	16	17
8	Ed. Jose Gabriel Maldonado	18	16	34
9	Ed. Jose Rafael Arboleda	16	0	16
10	Ed. Julio Carrizosa	3	0	3
11	Ed. Pablo VI	16	0	16
12	Facultad de Artes	1	16	17
13	Hospital Universitario San Ignacio	17	15	32
14	TOTALES	96	80	176