Booklet

Data Engineering -
Nivelación/Práctica

¡Bienvenido/as!

Este Booklet (cuadernillo de ejercicios) ha sido creado para **ejercitar sobre los contenidos básicos necesarios para cursar Data Engineering.**

Este documento está compuesto por diferentes actividades con sus respectivas resoluciones, dividido por las herramientas básicas que se usarán para este curso:

- ✓ Python
- ✓ SQL (Base de datos)

Les sugerimos que **de no saber la resolución a los presentes ejercicios, conversen con su tutor sobre las posibilidades de elegir un curso acorde para seguir con su desarrollo dentro del mundo de la Data.**

En el siguiente slide se detalla la estructura de este cuadernillo, diseñado para fomentar su proceso de aprendizaje.

Cualquier duda que surja, sería buena idea conversarla con su tutor.

CODERHOUSE

Estructura

Elementos que componen el Booklet:

- ✓ **Actividades:** son propuestas de ejercitación práctica, basadas en problemáticas comunes. Algunas consignas son específicas, pero otras son más abiertas.

- ✓ **Resoluciones propuestas:** el booklet viene con resoluciones para comparar con sus respuestas. Las resoluciones son propuestas; hay varias soluciones posibles. Es por esto que su resolución puede presentar una funcionalidad similar o mejor a la propuesta en el booklet.

- ✓ **Scripts Python:** por cada actividad de Python se adjuntará una captura de pantalla de una resolución para comparar con tu resolución.

- ✓ **Scripts SQL:** por cada actividad de SQL, habrá un script de referencia, con el cual comparar si el código elaborado es similar a la funcionalidad esperada. La solución de los ejercicios estará basada en la siguiente base de datos.

Python

- ✓ Ejercicios
- ✓ Resoluciones

SQL

- ✓ Ejercicios
- ✓ Resoluciones

CODERHOUSE

Python

Ejercicios
Python

CONSIGNA

Ejercicio 1

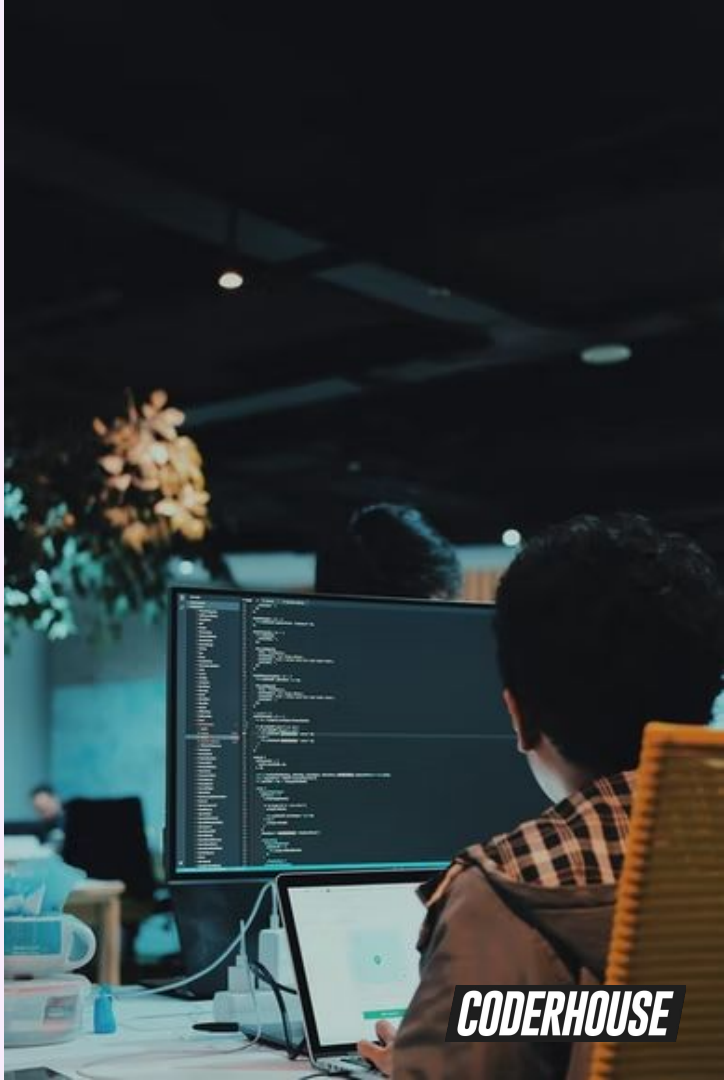
Escribir un programa que lea un número impar por teclado. Si el usuario no introduce un número impar, debe repetirse el proceso hasta que lo introduzca correctamente.

Referencia de imagen: Procreator UX Design Studio

CONSIGNA

Ejercicio 2

Escribir un programa que pida al usuario cuántos números quiere introducir. Luego que lea todos los números y realice una media aritmética.



Referencia de imagen: Arif Riyanto

CONSIGNA

Ejercicio 3

Utilizando la función range() y la conversión a listas generar las siguientes listas dinámicamente:

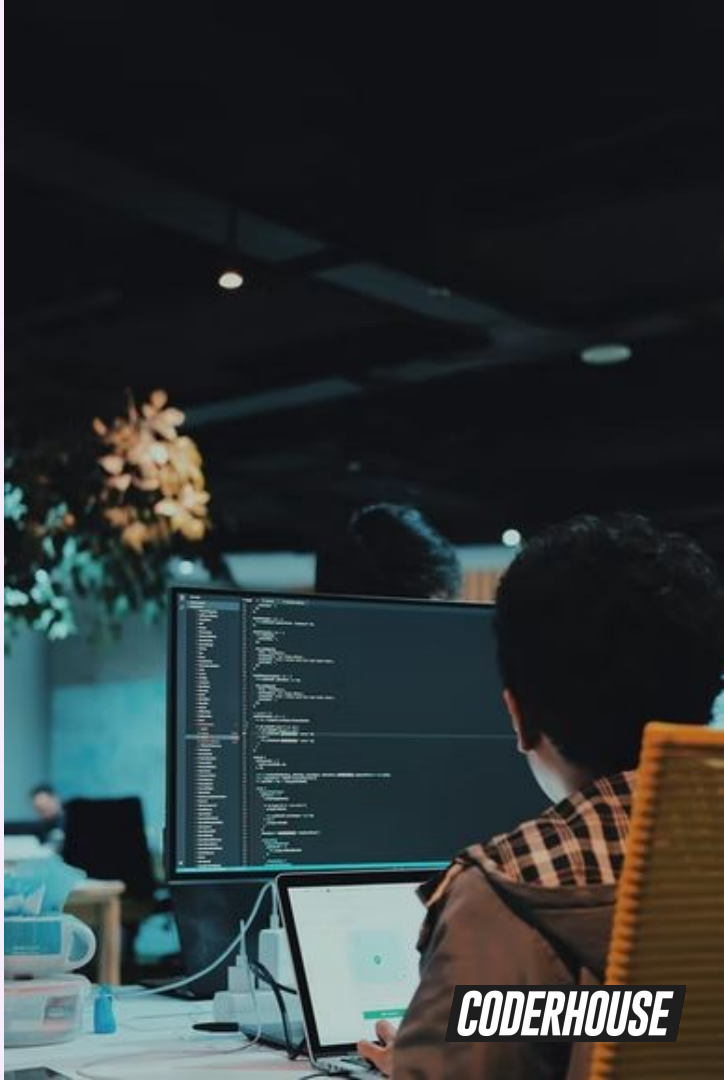
- ✓ Todos los números del 0 al 10 [0, 1, 2, ..., 10]
- ✓ Todos los números del -10 al 0 [-10, -9, -8, ..., 0]
- ✓ Todos los números pares del 0 al 20 [0, 2, 4, ..., 20]
- ✓ Todos los números impares entre -20 y 0 [-19, -17, -15, ..., -1]
- ✓ Todos los números múltiples de 5 del 0 al 50 [0, 5, 10, ..., 50]

Referencia de imagen: Procreator UX Design Studio

CONSIGNA

Ejercicio 4

Dadas dos listas (las que se quiera crear), generar una tercera con los elementos que estén presentes en AMBAS listas. Retornar esta nueva lista pero sin elementos duplicados.



Referencia de imagen: Arif Riyanto

CONSIGNA

Ejercicio 5

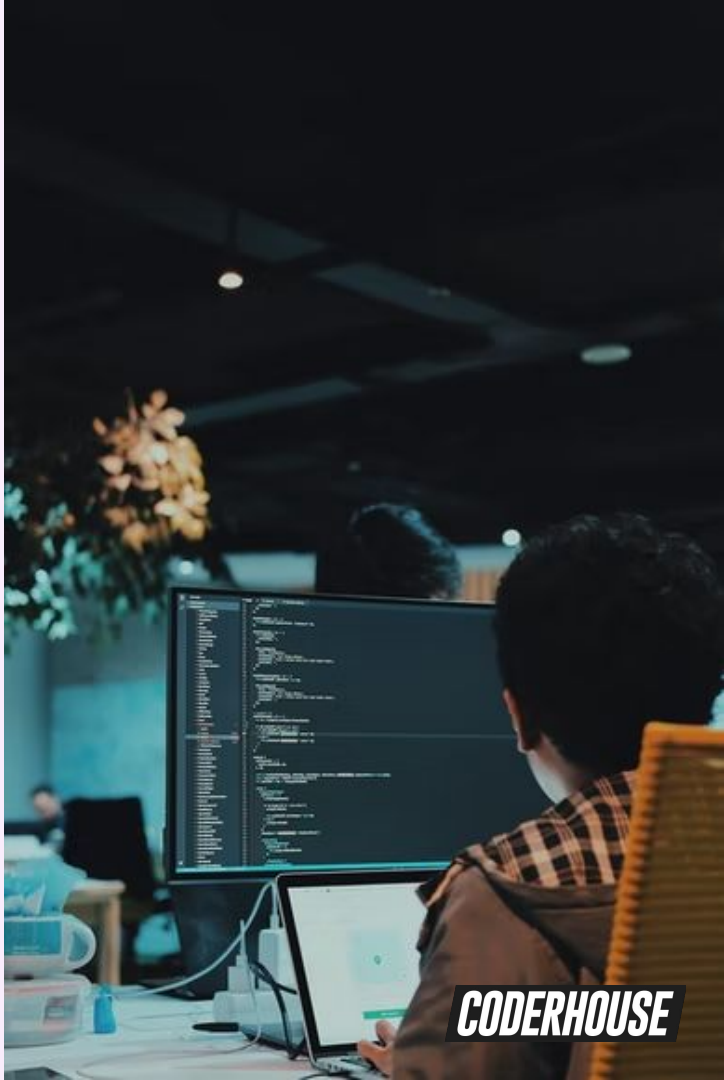
Escribir un programa que sume todos los números enteros impares desde el 0 hasta el 100

Referencia de imagen: Procreator UX Design Studio

CONSIGNA

Ejercicio 6

Contar cuantas veces aparece un elemento en una lista



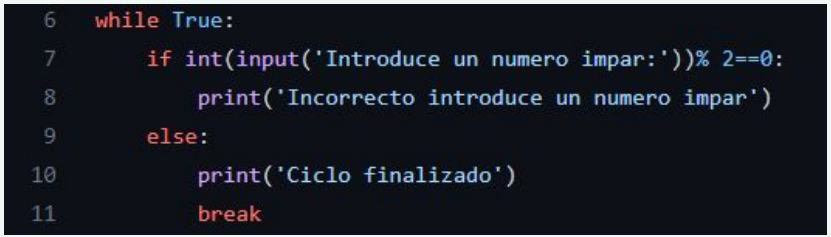
Referencia de imagen: Arif Riyanto

Resoluciones
Python

RESOLUCIÓN

Ejercicio 1

```
while True:
    if int(input('Introduce un numero impar:'))% 2==0:
        print('Incorrecto introduce un numero impar')
    else:
        print('Ciclo finalizado')
        break
```



RESOLUCIÓN

Ejercicio 2

```
cantidad=int(input('Introduce una cantidad de numeros
para sacar la media:'))

lista=[]

for i in range(cantidad):

    a=float(input('Introduce el numero {}:'.format(i+1)))

    lista.append(a)

print('La media de los numeros es:', sum(lista)/len(lista))
```

RESOLUCIÓN

Ejercicio 2

```
5  cantidad=int(input('Introduce una cantidad de numeros para sacar la media:'))
6  lista=[]
7  for i in range(cantidad):
8      a=float(input('Introduce el numero {}:'.format(i+1)))
9      lista.append(a)
10 print('La media de los numeros es:', sum(lista)/len(lista))
```

RESOLUCIÓN

Ejercicio 3

print(list(range(0,10+1,1)));

print([x for x in range(-10, 1)])

print([x for x in range(0,20+1,1) if x%2==0])

print([x for x in range(-19, -1+1)])

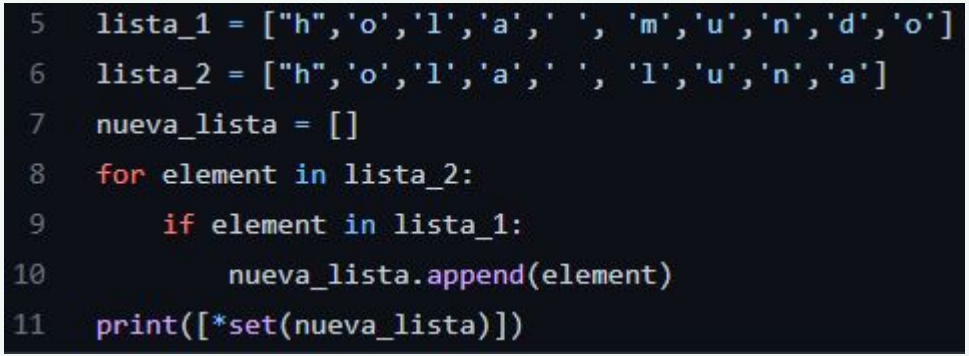
print([x for x in range(0,50+1,1) if x%5==0])

```
11  print(list(range(0,10+1,1)));
12  print([x for x in range(-10, 1)])
13  print([x for x in range(0,20+1,1) if x%2==0])
14  print([x for x in range(-19, -1+1)])
15  print([x for x in range(0,50+1,1) if x%5==0])
```

RESOLUCIÓN

Ejercicio 4

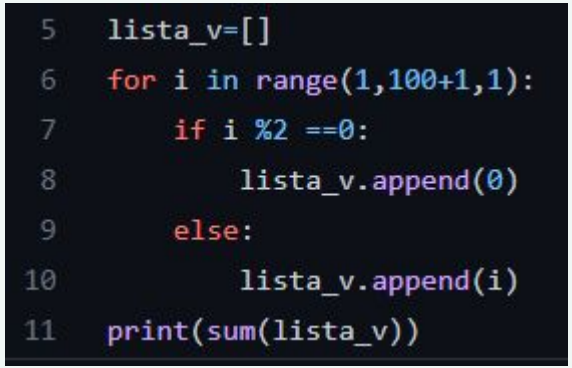
```
lista_1 = ["h",'o','l','a',' ', 'm','u','n','d','o']
lista_2 = ["h",'o','l','a',' ', 'l','u','n','a']
nueva_lista = []
for element in lista_2:
    if element in lista_1:
        nueva_lista.append(element)
print([*set(nueva_lista)])
```



RESOLUCIÓN

Ejercicio 5

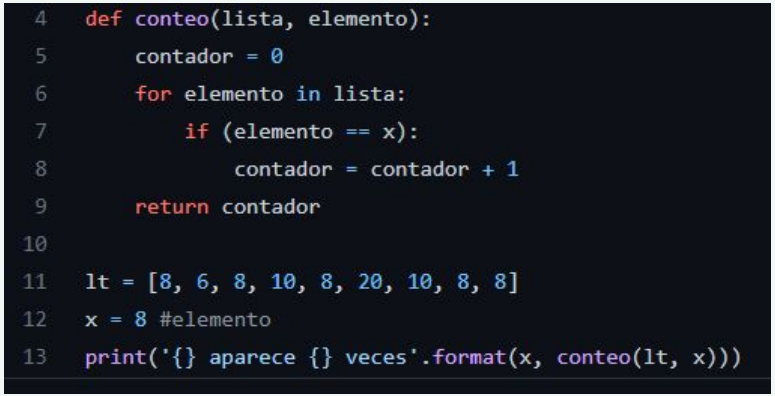
```
lista_v=[]
for i in range(1,100+1,1):
    if i %2 ==0:
        lista_v.append(0)
    else:
        lista_v.append(i)
print(sum(lista_v))
```



RESOLUCIÓN

Ejercicio 6

```
def conteo(lista, elemento):
    contador = 0
    for elemento in lista:
        if (elemento == x):
            contador = contador + 1
    return contador
lt = [8, 6, 8, 10, 8, 20, 10, 8, 8]
x = 8 #elemento
print('{} aparece {} veces'.format(x, conteo(lt, x)))
```



SQL

Ejercicios
SQL

CONSIGNA

Ejercicio 1

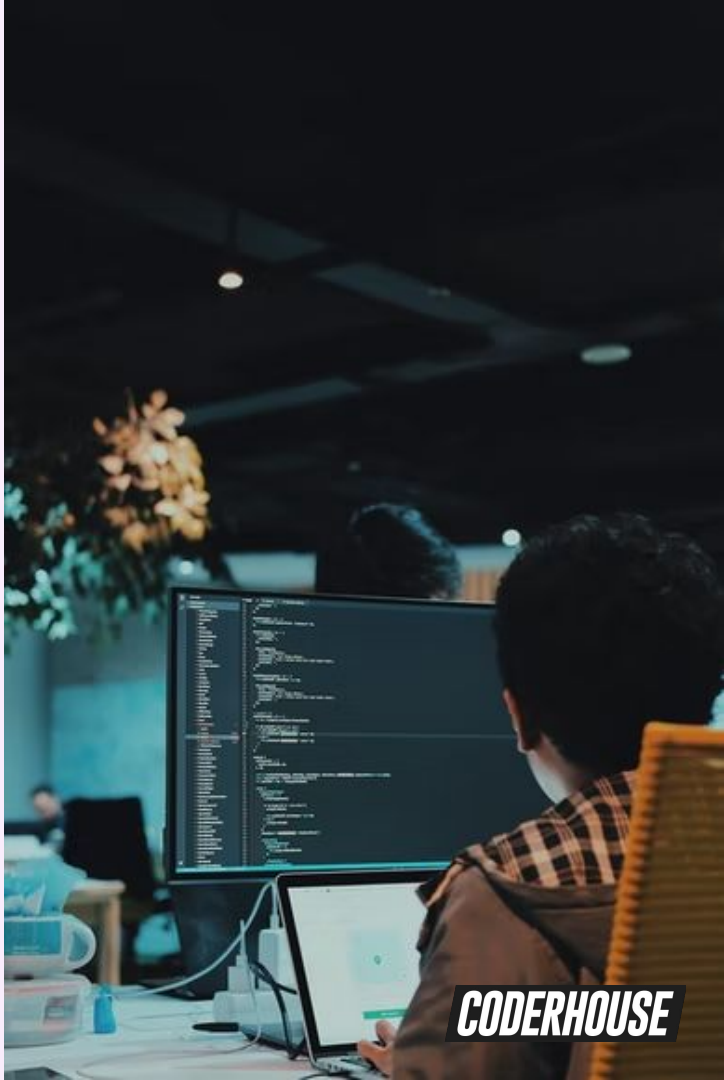
De la base de datos dada. Extraer agentes cuyo nombre empieza por M o termina en O.

Referencia de imagen: Procreator UX Design Studio

CONSIGNA

Ejercicio 2

De la base de datos dada. Escribir una consulta que produzca una lista, en orden alfabético, de todas las distintas ocupaciones en la tabla Customer que contengan la palabra "Engineer".



Referencia de imagen: Arif Riyanto

CONSIGNA

Ejercicio 3

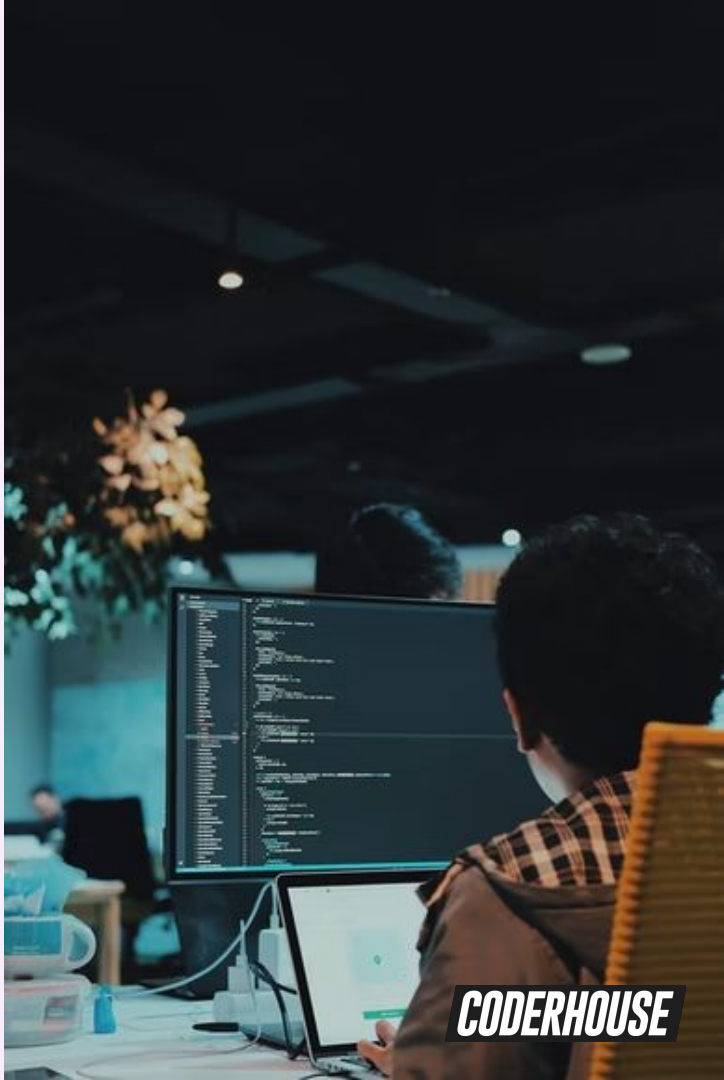
De la base de datos dada. Escribir una consulta que devuelva el ID del cliente, su nombre y una columna nueva llamada “Mayor30” que contenga "Sí" si el cliente tiene más de 30 años y "No" en caso contrario.

Referencia de imagen: Procreator UX Design Studio

CONSIGNA

Ejercicio 4

De la base de datos dada. Escribir una consulta que devuelva todas las llamadas realizadas a clientes de la profesión de ingeniería y muestre si son mayores o menores de 30, así como si terminaron comprando el producto de esa llamada.



Referencia de imagen: Arif Riyanto

CONSIGNA

Ejercicio 5

De la base de datos dada. Escribir dos consultas:

1. Una que calcule las ventas totales y las llamadas totales realizadas a los clientes de la profesión de ingeniería.
2. Otra que calcule las mismas métricas para toda la base de clientes.

Referencia de imagen: Procreator UX Design Studio

CONSIGNA

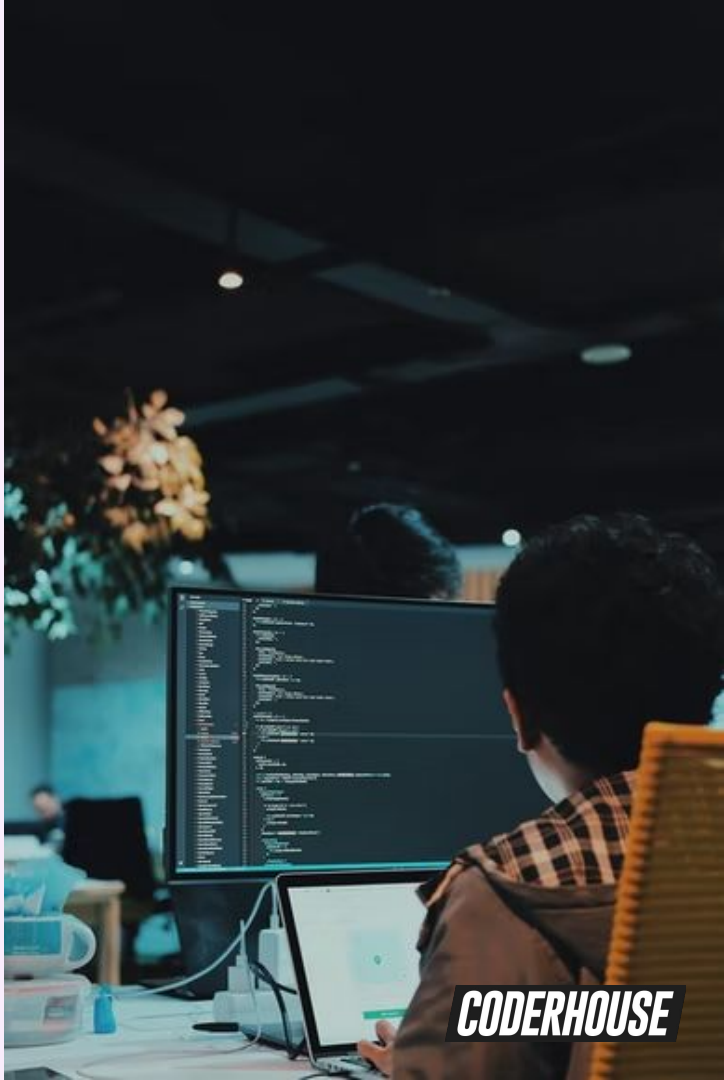
Ejercicio 6

De la base de datos dada. Escribir una consulta que devuelva

- ✓ **Para cada agente:** el nombre del agente, la cantidad de llamadas, las llamadas más largas y más cortas, la duración promedio de las llamadas y la cantidad total de productos vendidos.
- ✓ **Nombra las columnas:** AgentName, NCalls, Shortest, Longest, AvgDuration y TotalSales
- ✓ **Luego ordenar la tabla por:** AgentName en orden alfabético.

Consejo: Asegurarse de incluir la cláusula WHERE PickedUp = 1 para calcular solo el promedio de todas las llamadas que fueron atendidas (de lo contrario ¡todas las duraciones mínimas serán 0!)

Referencia de imagen: Arif Riyanto



CONSIGNA

Ejercicio 7

De la base de datos dada. Escribir una consulta que extraiga dos métricas del desempeño de los agentes de ventas que le interesan a su empresa:

1. Para cada agente, cuántos segundos en promedio les toma vender un producto cuando tienen éxito.
2. Para cada agente, cuántos segundos en promedio permanecen en el teléfono antes de darse por vencidos cuando no tienen éxito.

Referencia de imagen: Procreator UX Design Studio

Resoluciones
SQL

RESOLUCIÓN

Ejercicio 1

select * from agents

where name like 'M%' or name like '%o'

```
4   select * from agents
5   where name like 'M%' or name like '%o'
```

RESOLUCIÓN

Ejercicio 2

SELECT DISTINCT Occupation

FROM customers

WHERE Occupation LIKE '%Engineer%'

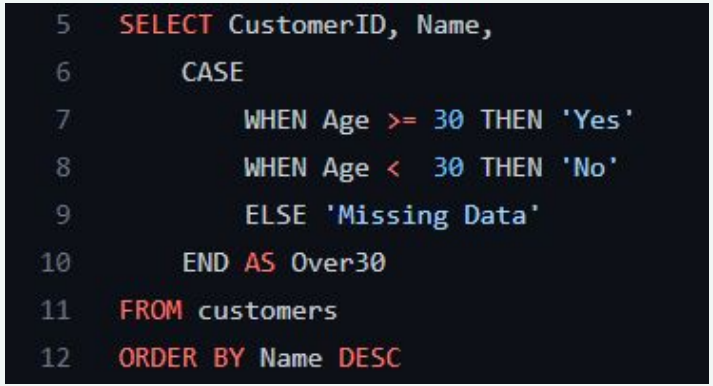
ORDER BY Occupation

```
6   SELECT DISTINCT Occupation
7   FROM customers
8   WHERE Occupation LIKE '%Engineer%'
9   ORDER BY Occupation
```

RESOLUCIÓN

Ejercicio 3

```
SELECT CustomerID, Name,
    CASE
        WHEN Age >= 30 THEN 'Yes'
        WHEN Age <  30 THEN 'No'
        ELSE 'Missing Data'
    END AS Over30
FROM customers
ORDER BY Name DESC
```

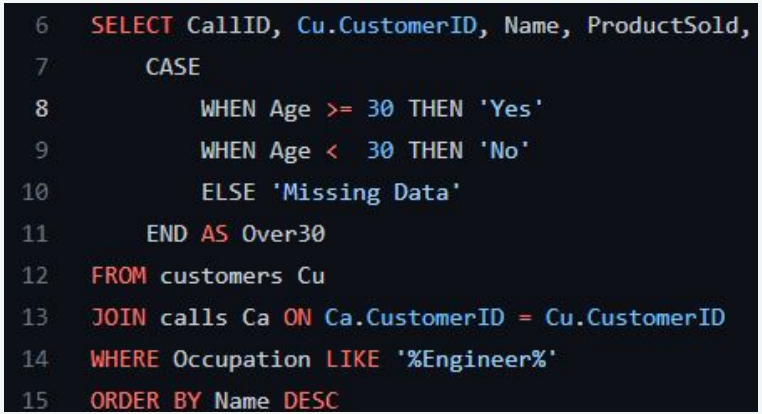


CODERHOUSE

RESOLUCIÓN

Ejercicio 4

```
SELECT CallID, Cu.CustomerID, Name, ProductSold,
    CASE
        WHEN Age >= 30 THEN 'Yes'
        WHEN Age <  30 THEN 'No'
        ELSE 'Missing Data'
    END AS Over30
FROM customers Cu
JOIN calls Ca ON Ca.CustomerID = Cu.CustomerID
WHERE Occupation LIKE '%Engineer%'
ORDER BY Name DESC
```



CODERHOUSE

RESOLUCIÓN

Ejercicio 5

SELECT SUM(ProductSold) AS TotalSales,
COUNT(*) AS NCalls

FROM customers Cu

JOIN calls Ca ON Ca.CustomerID =
Cu.CustomerID

WHERE Occupation LIKE '%Engineer%'

```
6  SELECT SUM(ProductSold) AS TotalSales, COUNT(*) AS NCalls
7  FROM customers Cu
8  JOIN calls Ca ON Ca.CustomerID = Cu.CustomerID
9  WHERE Occupation LIKE '%Engineer%'
```

RESOLUCIÓN

Ejercicio 6

```
SELECT Name AS AgentName, COUNT(*) AS NCalls, MIN(Duration) AS Shortest, MAX(Duration) AS
Longest, ROUND(AVG(Duration),2) AS AvgDuration, SUM(ProductSold) AS TotalSales

FROM calls C

    JOIN agents A ON C.AgentID = A.AgentID

WHERE PickeDup = 1

GROUP BY Name

ORDER BY Name
```

RESOLUCIÓN

Ejercicio 6

```
9   SELECT Name AS AgentName, COUNT(*) AS NCalls, MIN(Duration) AS Shortest, MAX(Duration) AS Longest, ROUND(AVG(Duration),2) AS AvgDuration, SUM(ProductSold) AS TotalSales
10  FROM calls C
11      JOIN agents A ON C.AgentID = A.AgentID
12  WHERE PickeDup = 1
13  GROUP BY Name
14  ORDER BY Name
```

RESOLUCIÓN

Ejercicio 7

```
SELECT a.name,
SUM(
  CASE
    WHEN productsold = 0 THEN duration
    ELSE 0
  END)/SUM(
  CASE
    WHEN productsold = 0 THEN 1
    ELSE 0
  END)
AS avgWhenNotSold ,
SUM(
  CASE
    WHEN productsold = 1 THEN duration
    ELSE 0
  END)/SUM(
    CASE WHEN productsold = 1 THEN 1
    ELSE 0
  END)
```

Continuación del código

```
AS avgWhenSold
FROM calls c
JOIN agents a ON c.agentid = a.agentid
GROUP BY a.name
ORDER BY 1
```

RESOLUCIÓN

Ejercicio 7

```sql
SELECT a.name,
SUM(
   CASE
       WHEN productsold = 0 THEN duration
       ELSE 0
   END)/SUM(
   CASE
       WHEN productsold = 0 THEN 1
       ELSE 0
   END)
AS avgWhenNotSold ,
SUM(
   CASE
       WHEN productsold = 1 THEN duration
       ELSE 0
   END)/SUM(
       CASE WHEN productsold = 1 THEN 1
       ELSE 0
   END)
AS avgWhenSold
FROM calls c
JOIN agents a ON c.agentid = a.agentid
GROUP BY a.name
ORDER BY 1
```

FINAL DE LOS EJERCICIOS

¡Lo lograron!

Si llegaron hasta aquí y lograron realizar cada uno de los ejercicios propuestos, déjennos decirles que ¡You rock!

Esperamos que toda la información brindada haya sido de mucho provecho y que sirva como fundamento o para refrescar la memoria sobre conceptos básicos para cursar Data Engineering.

Cualquier duda que haya surgido, recomendamos conversarla con su tutor. De ver dificultades con los ejercicios iniciales considerar pedir recomendaciones de otros cursos.

Como consejo para su participación en este curso, les recordamos que nos comprometemos a darles las mejores didácticas y técnicas pedagógicas al alcance, pero que es fundamental el compromiso de cada uno de ustedes consigo mismo, **la clave del éxito no reside en el talento sino en la constancia**. Sigan practicando todo lo que se les enseñe y vayan por ese potencial que pueden lograr.

¡Muchos éxitos en todo lo que se propongan!