Ejercicios SQL Bootcamp Data Engineer - EDVAI

Consignas:

A) Escribir las querys/consultas necesarias para llegar al resultado (print), usando windows functions.
B) Las consultas deben ser subidas a un proyecto público de github y compartir el link al instructor.

Nota: el proyecto de github debe tener al menos dos commits (puede ser uno por el punto B y otro subir un archivo .sql con las consultas) y deberá ser compartido con el instructor.

AVG

1. Obtener el promedio de precios por cada categoría de producto. La cláusula OVER(PARTITION BY CategoryID) específica que se debe calcular el promedio de precios por cada valor único de CategoryID en la tabla.

```
select c.category_name, p.product_name, p.unit_price,
  avg(p.unit_price) over (partition by p.category_id) as averagebycategory
from products p
  inner join categories c
  on p.category_id=c.category_id
```

2. Obtener el promedio de venta de cada cliente

```
select
  avg(op.order_price) over (partition by o.customer_id) as avgorderamount,
  o.*
from orders o
  inner join (
    select
      od2.order_id,
      sum(od2.unit_price * od2.quantity *(1-od2.discount)) as order_price
      from
        order_details od2 group by od2.order_id
  ) as op
  on o.order_id = op.order_id
```

3. Obtener el promedio de cantidad de productos vendidos por categoría (product_name, quantity_per_unit, unit_price, quantity, avgquantity) y ordenarlo por nombre de la categoría y nombre del producto

```
select
  p.product_name,
  c.category_name,
  p.quantity_per_unit,
  p.unit_price,
  od.quantity,
  avg(od.quantity) over (partition by p.category_id) as avgquantity
from order_details od
  inner join products p
  on od.product_id=p.product_id
  inner join categories c
  on p.category_id=c.category_id
order by c.category_name, p.product_name
```

MIN

4. Selecciona el ID del cliente, la fecha de la orden y la fecha más antigua de la orden para cada cliente de la tabla 'Orders'.

```
select o.customer_id, o.order_date,
  min(o.order_date) over (partition by o.customer_id) as earliestorderdate
from orders o
```

MAX

5. Seleccione el id de producto, el nombre de producto, el precio unitario, el id de categoría y el precio unitario máximo para cada categoría de la tabla Products.

```
select p.product_id, p.product_name, p.unit_price, p.category_id,
  max(p.unit_price) over (partition by p.category_id) as maxunitprice
from products p
```

Row_number

6. Obtener el ranking de los productos más vendidos

```
select
  ROW_NUMBER() over (order by totals.totalquantity desc),
  p.product_name,
  totals.totalquantity
from products p
  inner join (select
    od.product_id,
    sum(od.quantity) as totalquantity
    from order_details od
    group by od.product_id) as totals
  on p.product_id=totals.product_id
```

7. Asignar números de fila para cada cliente, ordenados por customer_id

```
 select
  ROW_NUMBER() over (order by totals.totalquantity desc),
  p.product_name,
  totals.totalquantity
from products p
  inner join (select
    od.product_id,
    sum(od.quantity) as totalquantity
    from order_details od
    group by od.product_id) as totals
  on p.product_id=totals.product_id
```

8. Obtener el ranking de los empleados más jóvenes () ranking, nombre y apellido del empleado, fecha de nacimiento)

```
select
  row_number() over (order by e.birth_date desc),
  concat(e.first_name,' ', e.last_name) as employeename,
  e.birth_date
from employees e
```

SUM

9. Obtener la suma de venta de cada cliente

```
select
  sum(od.unit_price*od.quantity*(1-od.discount)) over (partition by
o.customer_id) as sumorderamount,
  o.*
from orders o
  inner join order_details od
  on o.order_id=od.order_id
```

10.Obtener la suma total de ventas por categoría de producto

```
select
  c.category_name,
  p.product_name,
  p.unit_price,
  p.unit_price,
  od.quantity,
  sum(od.unit_price*od.quantity) over (partition by p.category_id) as totalsales
from order_details od
  inner join products p
  on od.product_id=p.product_id
  inner join categories c
  on p.category_id=c.category_id
order by c.category_name, p.product_name
```

11. Calcular la suma total de gastos de envío por país de destino, luego ordenarlo por país y por orden de manera ascendente

```
select
  o.ship_country as country, o.order_id, o.shipped_date, o.freight,
  sum(o.freight) over (partition by o.ship_country) as totalshippingcosts
from orders o
 order by o.ship_country, o.order_id
```

RANK

12.Ranking de ventas por cliente

```
select distinct
  o.customer_id, c.company_name, "Total Sales",
  dense_rank() over (order by "Total Sales" desc) as Rank
from orders o
inner join customers c
  on o.customer_id=c.customer_id
inner join (
  select o.customer_id,
    sum(orderprices.orderprice) as "Total Sales"
  from orders o
  inner join (
    select
      od.order_id,
      sum(od.product_id*od.unit_price*(1-od.discount)) as orderprice
    from order_details od
    group by od.order_id
    ) as orderprices
  on o.order_id=orderprices.order_id
  group by o.customer_id
) as Totals
on o.customer_id=Totals.customer_id
order by Rank
```

13.Ranking de empleados por fecha de contratación

```
select e.employee_id, e.first_name, e.last_name, e.hire_date,
  rank() over (order by e.hire_date)
from employees e
```

14.Ranking de productos por precio unitario

```
select p.product_id, p.product_name, p.unit_price,
  rank() over (order by p.unit_price desc)
from products p
```

LAG

15.Mostrar por cada producto de una orden, la cantidad vendida y la cantidad vendida del producto previo.

```
with q as
(
  select od.order_id, p.product_id,
    sum(od.quantity) as quantity
  from products p
  inner join order_details od
  on p.product_id=od.product_id
  group by od.order_id, p.product_id
)
select *,
  lag(q.quantity) over (order by q.order_id) as prevquantity
from q;
```

16.Obtener un listado de ordenes mostrando el id de la orden, fecha de orden, id del cliente y última fecha de orden.

```
select o.order_id, o.order_date, o.customer_id,
lag(o.order_date) over (partition by o.customer_id order by o.order_date)
from orders o
order by o.customer_id
```

17.Obtener un listado de productos que contengan: id de producto, nombre del producto, precio unitario, precio del producto anterior, diferencia entre el precio del producto y precio del producto anterior.

```
select p.product_id, p.product_name, p.unit_price,
  lag(p.unit_price) over (order by p.product_id) as lastunitprice,
  p.unit_price - lag(p.unit_price) over (order by p.product_id) as
pricedifference
from products p
```

LEAD

18.Obtener un listado que muestra el precio de un producto junto con el precio del producto siguiente:

```
select p.product_name, p.unit_price,
  lead(p.unit_price) over (order by p.product_id) as nextprice
from products p
```

19.Obtener un listado que muestra el total de ventas por categoría de producto junto con el total de ventas de la categoría siguiente

```
with sales as(
  select c.category_name,
    sum(od.unit_price*od.quantity) as totalsales
  from order_details od
    inner join products p
    on od.product_id=p.product_id
    inner join categories c
    on p.category_id=c.category_id
  group by c.category_name
)
select *,
  lead(s.totalsales) over (order by s.category_name) as nexttotalsales
from sales as s;
```