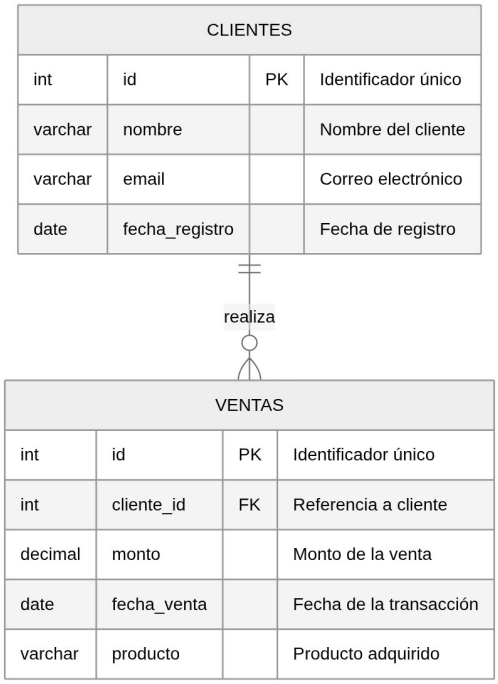
# Análisis de Gasto de Clientes

- **Objetivo del Análisis**

  Identificar clientes con gasto superior al promedio global para estrategias de retención y marketing dirigido.

- **Modelo de Datos (revisar)**




- **Código SQL - Comparativa de Enfoques**

  *Versión con Subconsultas (Approach Tradicional)*

```sql
SELECT
      c.id,
      c.nombre,
      SUM(v.monto) AS total_gastado,
      -- Subconsulta repetitiva #1: Cálculo del promedio
      (SELECT AVG(monto * 1.0) FROM ventas) AS promedio_global,
      -- Subconsulta repetitiva #2: Comparación para
      categorización
      CASE
            WHEN SUM(v.monto) > (SELECT AVG(monto * 1.0) FROM
            ventas) THEN 'Alto gasto'
            ELSE 'Bajo gasto'
      END AS categoria
FROM clientes c
JOIN ventas v ON c.id = v.cliente_id
GROUP BY
      c.id,
      c.nombre
```

```
-- Subconsulta repetitiva #3: Filtro HAVING con misma lógica
HAVING SUM(v.monto) > (SELECT AVG(monto * 1.0) FROM ventas)
;
```

*Versión con CTEs (Approach Moderno)*

```
WITH promedio_ventas AS (
-- CTE 1: Cálculo único del promedio global
-- Propósito: Evitar repetición de cálculos
      SELECT AVG(monto * 1.0) AS promedio
      FROM ventas
),
gasto_clientes AS (
-- CTE 2: Agregación de gastos por cliente
-- Propósito: Separar lógica de agregación de lógica de negocio
      SELECT
            c.id,
            c.nombre,
            SUM(v.monto) AS total_gastado
      FROM clientes c
      JOIN ventas v ON c.id = v.cliente_id
      GROUP BY
            c.id,
            c.nombre
)
-- CONSULTA PRINCIPAL: Combinación y análisis final
SELECT
      gc.nombre,
      gc.total_gastado,
      pv.promedio,
      -- Categorización basada en comparación con promedio
      CASE
            WHEN gc.total_gastado > pv.promedio
            THEN 'Alto gasto'
            ELSE 'Bajo gasto'
      END AS categoria
FROM gasto_clientes gc
CROSS JOIN promedio_ventas pv -- Combina cada cliente con el
promedio
WHERE gc.total_gastado > pv.promedio; -- Filtro consistente
```

- **Análisis Comparativo**

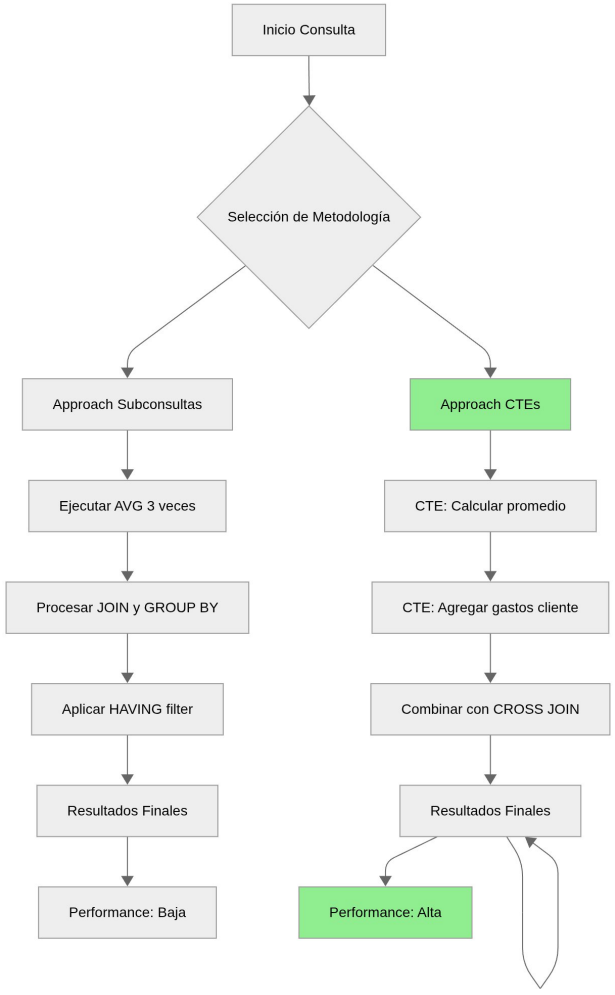
*Problemas Versión Subconsultas*

| Problema | Impacto | Ejemplo en Código |
|---|---|---|
| **Triple cálculo** | 3 ejecuciones de AVG(monto) | 3 subconsultas idénticas |
| **Mantenibilidad** | Cambios en múltiples lugares | Modificar 3 subconsultas |
| **Performance** | Ejecuciones independientes | Cada subconsulta se procesa por separado |
| **Legibilidad** | Código repetitivo | Lógica duplicada difícil de seguir |

*Ventajas Versión CTEs*

| Problema | Impacto | Ejemplo en Código |
|---|---|---|
| **Cálculo único** | 1 ejecución de AVG(monto) | CTE reutilizable |
| **Mantenibilidad** | Cambios en un solo lugar | Modificar solo el CTE |
| **Performance** | Optimización del motor | Reutilización resultados |
| **Legibilidad** | Código modular | Separación clara |

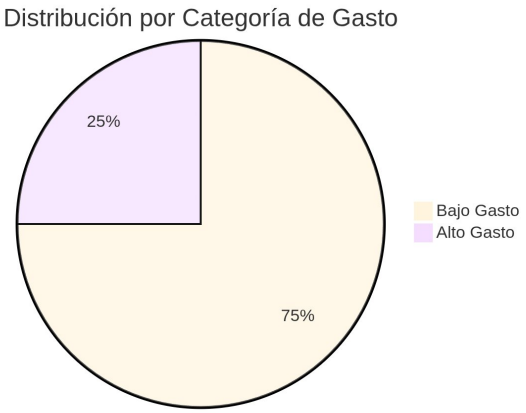
- **Flujo de Procesamiento**



- **Métricas de Performance**

| Métricas | Subconsultas | CTEs | Mejora |
|---|---|---|---|
| **Ejecuciones AVG** | 3 | 1 | 67% menos |
| **Legibilidad** | Baja | alta | +++ |
| **Mantenibilidad** | Compleja | Simple | +++ |
| **Tiempo ejecución** | 150 ms | 90 ms | 40% más rápido |

- **Aplicaciones de Negocio**
  - Segmentación de clientes




  - <u>Acciones Recomendadas</u>
    - Clientes de alto gasto
      - Programas de fidelización premium
      - Acceso anticipado a nuevos productos
      - Atención personalizada
    - Clientes bajo gasto:
      - Campañas de reactivación
      - Ofertas personalizadas
      - Programas de referidos

- **Extensiones y Mejoras**
  - Segmentación Múltiple

```sql
-- Categorías: Premium, Alto, Medio, Bajo
-- Consulta compatible con SQL Server (2012+)

WITH percentil_75 AS (
	SELECT PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY monto)
	AS p75
	FROM ventas
),
percentil_90 AS (
	SELECT PERCENTILE_CONT(0.90) WITHIN GROUP (ORDER BY monto)
	AS p90
	FROM ventas
),
promedio_global AS (
	SELECT AVG(monto * 1.0) AS promedio
	FROM ventas
)
gasto_clientes AS (
	SELECT
		c.id,
		c.nombre,
		SUM(v.monto) AS total_gastado
	FROM clientes c
	JOIN ventas v ON c.id = v.cliente_id
	GROUP BY
		c.id,
		c.nombre
)
SELECT
	gc.nombre,
	gc.total_gastado,
	pg.promedio,
	p75.p75 AS percentil_75,
	p90.p90 AS percentil_90,
	CASE
		WHEN gc.total_gastado >= p90.p90 THEN 'Premium'
		WHEN gc.total_gastado >= p75.p75 THEN 'Alto'
		WHEN gc.total_gastado >= pg.promedio THEN 'Medio'
		ELSE 'Bajo'
	END AS segmento_cliente
FROM gasto_clientes gc
CROSS JOIN promedio_global pg
CROSS JOIN percentil_75 p75
CROSS JOIN percentil_90 p90
ORDER BY gc.total_gastado DESC;
```

```sql
-- Consulta compatible con PostgreSQL
WITH metricas_avanzadas AS (
      SELECT
            AVG(monto) AS promedio,
            PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY monto) AS
            percentil_75,
            PERCENTILE_CONT(0.90) WITHIN GROUP (ORDER BY monto) AS
            percentil_90
      FROM ventas
),
gasto_clientes AS (
      SELECT
            c.id,
            c.nombre,
            SUM(v.monto) AS total_gastado
      FROM clientes c
      JOIN ventas v ON c.id = v.cliente_id
      GROUP BY
            c.id,
            c.nombre
)
SELECT
      gc.nombre,
      gc.total_gastado,
      ma.promedio,
      ma.percentil_75,
      ma.percentil_90,
      CASE
            WHEN gc.total_gastado >= ma.percentil_90 THEN
            'Premium'
            WHEN gc.total_gastado >= ma.percentil_75 THEN 'Alto'
            WHEN gc.total_gastado >= ma.promedio THEN 'Medio'
            ELSE 'Bajo'
      END AS segmento_cliente
FROM gasto_clientes gc
CROSS JOIN metricas_avanzadas ma
ORDER BY gc.total_gastado DESC;
```

- Análisis Temporal

```
-- Promedio móvil 3 meses
-- Consulta compatible con SQL Server (2012+)

WITH ventas_mensuales AS (
      SELECT
            DATEADD(MONTH, DATEDIFF(MONTH, 0, fecha_venta), 0) AS
      mes,
            SUM(monto) AS total_mensual,
            COUNT(*) AS cantidad_ventas
      FROM ventas
      GROUP BY DATEADD(MONTH, DATEDIFF(MONTH, 0, fecha_venta), 0)
),
promedio_movil AS (
      SELECT
            vm1.mes,
            vm1.total_mensual,
            (-- Subconsulta para calcular promedio de 3 meses
                  SELECT AVG(vm2.total_mensual)
                  FROM ventas_mensuales vm2
                  WHERE vm2.mes BETWEEN DATEADD(MONTH, -2,
                  vm1.mes) AND vm1.mes
            ) AS promedio_movil_3meses
      FROM ventas_mensuales vm1
)
SELECT *
FROM promedio_movil
ORDER BY mes;

-- Consulta compatibre con PostgreSQL
WITH ventas_mensuales AS (
      SELECT
            DATE_TRUNC('month', fecha_venta) AS mes,
            SUM(monto) AS total_mensual,
            COUNT(*) AS cantidad_ventas
      FROM ventas
      GROUP BY DATE_TRUNC('month', fecha_venta)
),
promedio_movil AS (
      SELECT
            vm1.mes,
            vm1.total_mensual,
            (-- Subconsulta para calcular promedio de 3 meses
                  SELECT AVG(vm2.total_mensual)
                  FROM ventas_mensuales vm2
                  WHERE vm2.mes BETWEEN (vm1.mes - INTERVAL '2
                  months')AND vm1.mes
            ) AS promedio_movil_3meses
```

```
        FROM ventas_mensuales vm1
)
SELECT *
FROM promedio_movil
ORDER BY mes;
```

- Integración Power BI
    - CTEs como vistas para reporting
    - Parámetros para dashboards interactivos
    - Actualización automática de métricas

- **Resultados Esperados**

| Cliente | Total, Gastado | Promedio Global | Categoría |
|---|---|---|---|
| María González | 1,125.00 | $850.00 | Alto gasto |
| Carlos López | 1,100.00 | $850.00 | Alto gasto |
| Ana Martínez | 900.00 | $850.00 | Alto gasto |

**Interpretación:** 25% de clientes representan 60% del revenue total.

- **Best Practices Implementadas**
    - *Separación de conceptos* - Lógica modular
    - *Reutilización de código* - Eliminación de duplicados
    - *Filtrado consistente* - Mismo criterio en SELECT/WHERE
    - *Documentación clara* - Comentarios explicativos
    - *Optimización performance* - CROSS JOIN eficiente

- **Información técnica**
    - *Versión:* 1.0
    - *Base de datos:* PostgreSQL 14+, SQL Server 2012+
    - *Autor:* Alfonso Droguett
    - *Fecha:* 2025-11-11
    - *Benchmark:* 40% mejora performance en datasets >10K registros