**1 - vista\_ConCubo\_2025**

CREATE OR ALTER VIEW vista\_ConCubo\_2025 AS

SELECT

ID,

TRY\_CAST(SUBSTRING(ID, PATINDEX('%[0-9]%', ID), LEN(ID)) AS INT) AS ID\_Limpio,

Renglon,

Estado,

-- Corregimos fechas restando 2 días (desfase histórico)

DATEADD(DAY, -2, TRY\_CAST(Inicio AS DATETIME)) AS Inicio\_Corregido,

DATEADD(DAY, -2, TRY\_CAST(Fin AS DATETIME)) AS Fin\_Corregido,

-- Formatos legibles para Power BI sin jerarquía automática

CONVERT(VARCHAR(16), DATEADD(DAY, -2, TRY\_CAST(Inicio AS DATETIME)), 120) AS Inicio\_Legible\_Texto,

CONVERT(VARCHAR(16), DATEADD(DAY, -2, TRY\_CAST(Fin AS DATETIME)), 120) AS Fin\_Legible\_Texto,

-- Fecha base para agrupación

CONVERT(DATE, DATEADD(DAY, -2, TRY\_CAST(Inicio AS DATETIME))) AS Fecha,

-- Cálculo de duración total

DATEDIFF(SECOND, TRY\_CAST(Inicio AS DATETIME), TRY\_CAST(Fin AS DATETIME)) / 3600.0 AS Total\_Horas,

-- Horas separadas por estado

CASE WHEN Estado = 'Producción' THEN DATEDIFF(SECOND, TRY\_CAST(Inicio AS DATETIME), TRY\_CAST(Fin AS DATETIME)) / 3600.0 ELSE 0 END AS Horas\_Produccion,

CASE WHEN Estado = 'Preparación' THEN DATEDIFF(SECOND, TRY\_CAST(Inicio AS DATETIME), TRY\_CAST(Fin AS DATETIME)) / 3600.0 ELSE 0 END AS Horas\_Preparacion,

CASE WHEN Estado = 'Maquina Parada' THEN DATEDIFF(SECOND, TRY\_CAST(Inicio AS DATETIME), TRY\_CAST(Fin AS DATETIME)) / 3600.0 ELSE 0 END AS Horas\_Parada,

CASE WHEN Estado = 'Mantenimiento' THEN DATEDIFF(SECOND, TRY\_CAST(Inicio AS DATETIME), TRY\_CAST(Fin AS DATETIME)) / 3600.0 ELSE 0 END AS Horas\_Mantenimiento,

-- Producción buena

TRY\_CAST(CantidadBuenosProducida AS FLOAT) AS CantidadBuenosProducida,

TRY\_CAST(CantidadMalosProducida AS FLOAT) AS CantidadMalosProducida

FROM ConCubo

WHERE

TRY\_CAST(Inicio AS DATETIME) >= '2025-01-01'

AND TRY\_CAST(Inicio AS DATETIME) < '2026-01-01'

AND ISNUMERIC(SUBSTRING(ID, PATINDEX('%[0-9]%', ID), LEN(ID))) = 1;

**2 - vista\_ConCubo\_2025\_Eventos**

CREATE OR ALTER VIEW vista\_ConCubo\_2025\_Eventos AS

SELECT

ID,

ID\_Limpio,

Renglon,

Estado,

Inicio\_Corregido,

Fin\_Corregido,

Inicio\_Legible\_Texto,

Fin\_Legible\_Texto,

CONVERT(DATE, Inicio\_Corregido) AS Fecha,

DATEDIFF(SECOND, Inicio\_Corregido, Fin\_Corregido) / 3600.0 AS Total\_Horas,

-- Horas separadas por estado (con corrección de Maquina Parada)

CASE WHEN Estado = 'Producción' THEN DATEDIFF(SECOND, Inicio\_Corregido, Fin\_Corregido) / 3600.0 ELSE 0 END AS Horas\_Produccion,

CASE WHEN Estado = 'Preparación' THEN DATEDIFF(SECOND, Inicio\_Corregido, Fin\_Corregido) / 3600.0 ELSE 0 END AS Horas\_Preparacion,

CASE WHEN Estado = 'Maquina Parada' THEN DATEDIFF(SECOND, Inicio\_Corregido, Fin\_Corregido) / 3600.0 ELSE 0 END AS Horas\_Parada,

CASE WHEN Estado = 'Mantenimiento' THEN DATEDIFF(SECOND, Inicio\_Corregido, Fin\_Corregido) / 3600.0 ELSE 0 END AS Horas\_Mantenimiento,

CantidadBuenosProducida,

CantidadMalosProducida

FROM vista\_ConCubo\_2025

WHERE YEAR(Inicio\_Corregido) = 2025;

**3 - vista\_ConCubo\_2025\_Eventos\_Diario**

CREATE OR ALTER VIEW vista\_ConCubo\_2025\_Eventos\_Diario AS

-- Primer CTE: agrega un número de secuencia por ID\_Limpio y máquina (Renglon)

WITH Base AS (

SELECT \*,

ROW\_NUMBER() OVER (

PARTITION BY ID\_Limpio, Renglon

ORDER BY Inicio\_Corregido ASC

) AS Nro\_Secuencia

FROM vista\_ConCubo\_2025\_Eventos

),

-- Segundo CTE: detecta inicio de bloque de preparación

PrepFlag AS (

SELECT \*,

CASE

WHEN Estado = 'Preparación' AND (

LAG(Estado) OVER (

PARTITION BY ID\_Limpio, Renglon

ORDER BY Inicio\_Corregido

) IS DISTINCT FROM 'Preparación'

) THEN 1

ELSE 0

END AS FlagPreparacion

FROM Base

),

-- Tercer CTE: genera la secuencia acumulativa de bloques de preparación

PrepSecuencia AS (

SELECT \*,

SUM(FlagPreparacion) OVER (

PARTITION BY ID\_Limpio, Renglon

ORDER BY Inicio\_Corregido

ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

) AS SecuenciaPreparacion

FROM PrepFlag

)

-- Resultado final

SELECT \*

FROM PrepSecuencia;

**4 - vista\_ConCubo\_2025\_Eventos\_Resumen\_Final**

CREATE OR ALTER VIEW vista\_ConCubo\_2025\_Resumen\_Final AS

SELECT

s.ID,

s.ID\_Limpio,

s.Renglon,

s.Estado,

s.Inicio\_Corregido,

s.Fin\_Corregido,

s.Inicio\_Legible\_Texto,

s.Fin\_Legible\_Texto,

s.Fecha,

s.Total\_Horas,

s.Horas\_Produccion,

s.Horas\_Preparacion,

s.Horas\_Parada, -- ✅ Ya corregida en vistas anteriores

s.Horas\_Mantenimiento,

s.CantidadBuenosProducida,

s.CantidadMalosProducida,

s.Nro\_Secuencia,

s.FlagPreparacion,

s.SecuenciaPreparacion,

VU.saccod1

FROM vista\_ConCubo\_2025\_Eventos\_Diario s

LEFT JOIN TablaVinculadaUNION VU

ON ISNUMERIC(VU.OP) = 1

AND TRY\_CAST(VU.OP AS INT) = s.ID\_Limpio;

**🧩 Resumen del flujo de vistas para el proyecto MEDORO – Año 2025**

**1. vista\_ConCubo\_2025**

🔹 **Objetivo:** Establecer una vista base limpia, corregida y confiable para los registros del año 2025.  
🔹 **Qué se hizo:**

* Se corrigió el desfase histórico de **2 días** en los campos Inicio y Fin.
* Se agregaron campos legibles en formato texto (Inicio\_Legible\_Texto, Fin\_Legible\_Texto) para evitar jerarquías automáticas en Power BI.
* Se calculó la **duración total del evento en horas** (Total\_Horas) y se desglosaron las horas según el tipo de Estado: Producción, Preparación, Mantenimiento y **Maquina Parada** (corrigiendo un error clave que antes usaba 'Parada').
* Se extrajeron las cantidades producidas (buenas y malas) y un ID\_Limpio para facilitar relaciones con otras tablas.

✅ **Resultado:** una vista sólida que sirve como base confiable para cualquier análisis posterior.

**2. vista\_ConCubo\_2025\_Eventos**

🔹 **Objetivo:** Reutilizar la vista base para tener una tabla de eventos ya corregidos, con cálculo de horas y sin filtrado adicional.  
🔹 **Qué se hizo:**

* Se eliminó el cálculo redundante.
* Se mantuvo el año 2025 como filtro explícito (WHERE YEAR(Inicio\_Corregido) = 2025).
* Se heredaron todos los campos corregidos desde la vista base.

✅ **Resultado:** tabla de eventos detallados por fila, lista para usar como insumo en transformaciones adicionales o análisis a nivel de evento.

**3. vista\_ConCubo\_2025\_Eventos\_Diario**

🔹 **Objetivo:** Identificar y numerar bloques únicos de preparación para evitar la **duplicación de tiempo** en el análisis.  
🔹 **Qué se hizo:**

* Se agregó una numeración secuencial (Nro\_Secuencia) por ID y máquina.
* Se identificó el inicio de cada bloque de **Preparación** usando un FlagPreparacion cuando el estado anterior no era preparación.
* Se construyó una **secuencia acumulativa** (SecuenciaPreparacion) para diferenciar múltiples bloques dentro de la misma orden.

✅ **Resultado:** permite distinguir con precisión cuándo comienza realmente una nueva preparación, clave para evitar **errores de duplicación** en KPIs y para medir eficiencia real.

**4. vista\_ConCubo\_2025\_Resumen\_Final**

🔹 **Objetivo:** Crear la vista final lista para conectarse a Power BI y combinar la información técnica con el dato de saccod1.  
🔹 **Qué se hizo:**

* Se arrastraron todos los campos calculados y secuencias desde vista\_ConCubo\_2025\_Eventos\_Diario.
* Se hizo una unión (LEFT JOIN) con la tabla TablaVinculadaUNION, extrayendo el código del sacabocado (saccod1) relacionado a cada OT.
* Se mantuvo el ID\_Limpio como clave de unión, con verificación de que OP fuera numérico.

✅ **Resultado:** tabla lista para análisis visual, con todo el contexto necesario (tiempos, secuencias, cantidades, sacabocados), sin errores de cálculo ni repeticiones.

**📊 Conclusión**

Con esta secuencia de vistas lograste:

* Corregir errores estructurales y de formato del sistema original.
* Evitar duplicaciones de tiempo de preparación.
* Calcular métricas fiables por estado.
* Preparar una estructura amigable para Power BI con claves limpias y campos legibles.
* Integrar datos técnicos con información de producción real.

El flujo está optimizado para análisis de eficiencia, tiempos muertos, control por máquina y validación cruzada con órdenes de trabajo.

IAN:

Veo muchas veces este cálculo

DATEDIFF(SECOND, ..., ...) / 3600.0

Para facilitar mantenimiento y evitar errores, podrías definir una CTE base con la duración ya calculada, o crear una función escalar dbo.HorasEntre(inicio, fin)

TRY\_CAST repetido varias veces

DATEADD(DAY, -2, TRY\_CAST(Inicio AS DATETIME))

Aparece varias veces por campo. Podés ahorrar parsing con un WITH inicial:

WITH DatosParseados AS (

SELECT \*,

TRY\_CAST(Inicio AS DATETIME) AS InicioDT,

TRY\_CAST(Fin AS DATETIME) AS FinDT

FROM ConCubo

)

Así luego usás directamente:

DATEADD(DAY, -2, InicioDT) AS Inicio\_Corregido

LEFT JOIN TablaVinculadaUNION VU

ON ISNUMERIC(VU.OP) = 1

AND TRY\_CAST(VU.OP AS INT) = s.ID\_Limpio

Esto anula el uso de índices. Si el campo VU.OP ya es numérico o se puede transformar en una columna adicional OP\_Int, sería mejor hacer:

-- En preparación

ALTER TABLE TablaVinculadaUNION ADD OP\_Int AS TRY\_CAST(OP AS INT) PERSISTED;

-- Y luego

ON VU.OP\_Int = s.ID\_Limpio

Cuidado con duplicación en joins si OP no es única

Asegurate de que OP en TablaVinculadaUNION tenga una única coincidencia por ID\_Limpio, o bien agregá un TOP 1, ROW\_NUMBER() o DISTINCT

esto en castellano

ISNUMERIC() y TRY\_CAST() desactivan índices (Un índice es una estructura auxiliar que crea el motor de base de datos para acelerar la búsqueda de datos en una tabla)

Cuando aplicás funciones en los campos del JOIN, el motor de SQL Server no puede usar índices. Eso significa escaneos completos de tabla (table scan), lo cual es lento si TablaVinculadaUNION tiene muchas filas.

Recalculás el cast en cada fila del JOIN

Si la tabla tiene 1 millón de filas, estás haciendo 1 millón de TRY\_CAST() durante el JOIN. Esto se puede evitar.

⚠️ Un JOIN que funciona… pero mata la performance.  
Lo resolvimos con una línea. Literal.  
  
Durante una mentoría 1:1 con **[Marcelo Fabián López Castro](https://www.linkedin.com/in/ACoAAAf-040BEsYdYnm-FqKgQElmtvADzOVbR8Y" \t "_self)**, estábamos revisando una cadena de vistas SQL para un dashboard de operaciones.  
  
Todo súper bien armado:  
 ✔️ Datos limpios,  
 ✔️ Fechas corregidas,  
 ✔️ Cálculos de eficiencia y tiempos muertos.  
  
Pero había algo raro: el dashboard tardaba en refrescar, incluso con solo 50.000 registros.  
  
Revisando juntos el código, encontré esto en uno de los JOIN:  
ON TRY\_CAST(tabla\_referencia.codigo AS INT) = **[tabla\_eventos.id](http://tabla_eventos.id/" \t "_self)**  
  
Marcelo había usado TRY\_CAST() directamente en el JOIN para asegurar que solo se compararan valores numéricos.  
  
Lógica válida, pero…  
  
Ese TRY\_CAST() desactiva cualquier índice que el motor de SQL pueda usar.  
  
📉 Resultado:  
Table scan completo  
Lento incluso con pocos registros  
No escalaba ni a palos  
  
Le sugerí una mejora simple y escalable:  
  
Crear una columna persistida ya casteada:  
ALTER TABLE tabla\_referencia  
ADD codigo\_int AS TRY\_CAST(codigo AS INT) PERSISTED;  
  
Reescribir el JOIN sin funciones:  
ON tabla\_referencia.codigo\_int = **[tabla\_eventos.id](http://tabla_eventos.id/" \t "_self)**  
  
✔️ Código más limpio  
⚡ Consulta mucho más rápida  
📈 Preparado para escalar sin romperse  
  
Muchos problemas de rendimiento en SQL no se resuelven con más infraestructura, sino con mejor criterio al escribir.  
  
¿Alguna vez tu dashboard se puso lento sin saber por qué?  
  
Contame en los comentarios o escribime.  
Pequeñas correcciones técnicas como esta hacen toda la diferencia a escala.

ACTUALIZACION 24/07/25

Perfecto Marcelo. Te armo ahora una **explicación clara, lista para tu reunión**, basada exclusivamente en esta vista:

**✅ Vista 1: vista\_ConCubo\_2025**

Esta vista es la base técnica sobre la cual se construyen todos los análisis posteriores del proyecto Medoro (versión 2025). Fue creada a partir de la tabla ConCubo, resolviendo errores históricos, estandarizando campos y preparando los datos para ser usados en Power BI o cualquier herramienta de análisis.

**🔧 Transformaciones clave aplicadas**

* **Corrección de fechas (Inicio y Fin)**  
  Se detectó un desfase de +2 días en los datos originales. Se corrige restando dos días con DATEADD(DAY, -2, ...).
* **Fechas legibles para Power BI**  
  Se generan versiones en texto (Inicio\_Legible\_Texto, Fin\_Legible\_Texto) para evitar que Power BI aplique jerarquías automáticas al importar fechas de tipo datetime.
* **Fecha de agrupación (Fecha)**  
  Campo exclusivo con solo año-mes-día (sin hora), ideal para analizar por día o semana.
* **Duración total del evento (Total\_Horas)**  
  Calculada en horas a partir de la diferencia entre Inicio y Fin.
* **Separación de tiempos por estado**  
  Cada fila clasifica el tiempo en:
  + Horas\_Produccion
  + Horas\_Preparacion
  + Horas\_Parada
  + Horas\_Mantenimiento  
    Esto permite sumar tiempos específicos sin filtros manuales.
* **Limpieza del campo ID**  
  Se extrae solo la parte numérica del campo ID (creando ID\_Limpio), eliminando textos como "Rotatek 700", que generaban errores de conversión.
* **Cantidad de producción**  
  Se incorporan CantidadBuenosProducida y CantidadMalosProducida como tipo FLOAT, si están disponibles.

**🗂 Filtrado por año**

La vista incluye exclusivamente registros del año **2025**, mediante este filtro:

WHERE

TRY\_CAST(Inicio AS DATETIME) >= '2025-01-01' AND

TRY\_CAST(Inicio AS DATETIME) < '2026-01-01'

Esto hace que:

* Se pueda replicar la vista para otros años fácilmente (solo cambiando las fechas).
* Los datos estén preparados para análisis anuales, comparativos y temporales.

**💡 ¿Qué resuelve esta vista?**

* Elimina errores de fechas que distorsionaban los análisis.
* Facilita la conexión a Power BI y la creación de visualizaciones limpias.
* Aísla el año 2025 para trabajar con un dataset cerrado y consistente.
* Permite sumar tiempos por tipo de estado sin re-procesar manualmente los datos.

**⚙️ ¿Cómo podrían incorporar años anteriores o futuros?**

1. Crear una nueva vista, por ejemplo vista\_ConCubo\_2024, con los mismos campos pero ajustando el filtro de fechas:
2. WHERE TRY\_CAST(Inicio AS DATETIME) >= '2024-01-01'
3. AND TRY\_CAST(Inicio AS DATETIME) < '2025-01-01'
4. O bien crear una **versión multianual** (por ejemplo, vista\_ConCubo\_TodosLosAnios) sin filtro de año, agregando el campo Ano con:
5. YEAR(TRY\_CAST(Inicio AS DATETIME)) AS Ano

Perfecto Marcelo. Vamos con la segunda vista. Te dejo la explicación detallada, lista para usar en tu README o reunión:

**✅ Vista 2: vista\_ConCubo\_2025\_Eventos**

Esta vista toma como base la vista anterior (vista\_ConCubo\_2025) y consolida los datos a nivel de **evento individual** ya corregido, manteniendo todos los atributos clave, pero simplificando la estructura para el análisis en Power BI.

**🔧 Transformaciones clave aplicadas**

* **Conserva campos ya corregidos y procesados**  
  Hereda directamente de vista\_ConCubo\_2025 todos los campos procesados previamente:
  + Inicio\_Corregido, Fin\_Corregido
  + ID\_Limpio
  + Tiempos por tipo de estado
  + Producción buena y mala
* **Agrega una columna de fecha (Fecha)**  
  Se crea Fecha con CONVERT(DATE, Inicio\_Corregido) para **agrupar eventos por día sin hora**.
* **Vuelve a calcular Total\_Horas y tiempos por estado**  
  Esto garantiza que los tiempos estén expresados correctamente a partir de los campos **ya corregidos**, sin depender del dato crudo original.
* **Filtra exclusivamente el año 2025**  
  El WHERE final se asegura de que solo entren registros cuyo Inicio\_Corregido pertenezca a 2025:
* WHERE YEAR(Inicio\_Corregido) = 2025

**📊 ¿Para qué se usa esta vista?**

Esta vista es ideal para:

* **Visualizar cada evento individual** (cada fila = un bloque de tiempo)
* Graficar **líneas de tiempo**, scatter plots o heatmaps por día y tipo de estado.
* Sumar tiempos por día, por orden (ID\_Limpio) o por tipo de evento.
* Usar como **fuente de tabla detallada en Power BI** para ver registros reales sin duplicaciones ni errores.

**💡 ¿Qué simplifica respecto a la anterior?**

* Elimina la lógica de conversión desde TRY\_CAST(Inicio) porque ya se heredó la fecha corregida.
* Evita que Power BI vuelva a interpretar mal las fechas.
* Permite **filtrar rápidamente** por año, fecha, ID, estado, etc.

**🧱 Relación con otras vistas**

* Es el **puente** entre la vista general (vista\_ConCubo\_2025) y los resúmenes por OT que vienen después.
* Se puede usar como tabla base para un **resumen por orden de trabajo, por día, o por máquina.**

Excelente, Marcelo. Esta vista es clave porque incorpora **lógica avanzada de secuencias**, algo que resolviste muy bien y que vale la pena explicar con claridad técnica. Acá va la descripción completa para tu README o para que lo expliques en la reunión:

**✅ Vista 3: vista\_ConCubo\_2025\_Eventos\_Diario**

Esta vista transforma los eventos individuales (vista\_ConCubo\_2025\_Eventos) en una estructura enriquecida con **bloques lógicos de preparación**, permitiendo un análisis **secuencial** de la producción. Utiliza funciones de ventana (ROW\_NUMBER, LAG, SUM OVER) para detectar y enumerar reinicios de la misma orden en una misma máquina.

**🔧 Transformaciones clave aplicadas**

**🔁 1. CTE Base: numeración por orden y máquina**

* Agrega un número de secuencia (Nro\_Secuencia) a cada fila, **ordenando cronológicamente** los eventos por combinación de ID\_Limpio y Renglon (la máquina).
* Esto permite ordenar correctamente los bloques de trabajo:
* ROW\_NUMBER() OVER (
* PARTITION BY ID\_Limpio, Renglon
* ORDER BY Inicio\_Corregido
* ) AS Nro\_Secuencia

**🔍 2. CTE PrepFlag: detección de inicios de bloques de preparación**

* Crea una nueva columna (FlagPreparacion) que vale **1 solo cuando comienza un nuevo bloque de preparación**.
* Usa LAG para comparar el estado actual con el anterior. Si el estado anterior **no era "Preparación"**, entonces estamos frente al inicio de un nuevo bloque:
* CASE
* WHEN Estado = 'Preparación' AND
* LAG(Estado) OVER (...) IS DISTINCT FROM 'Preparación'
* THEN 1 ELSE 0
* END AS FlagPreparacion

**🧮 3. CTE PrepSecuencia: asignación de secuencia acumulativa**

* Se suma acumulativamente la bandera FlagPreparacion, agrupando por ID\_Limpio y Renglon.
* Esto genera la columna SecuenciaPreparacion, que **identifica bloques consecutivos de preparación reales**, aún cuando sean múltiples por día.

**📊 ¿Para qué se usa esta vista?**

* Detecta correctamente **reinicios de la misma OT** en una misma máquina (por ejemplo, si una orden se interrumpió por otra, y luego vuelve).
* Permite analizar por **bloque de preparación real**, no solo por OT.
* Fundamental para crear indicadores como:
  + % de tiempo en modo preparación por bloque.
  + Detección de eventos ineficientes o recurrentes.
  + Secuencia cronológica validada de producción.

**💡 ¿Qué resuelve esta vista?**

* **Corrige un error clásico del sistema**: antes se asumía que una OT tenía solo un bloque de preparación, lo cual es falso.
* Esta vista **permite contar los bloques reales**, sin duplicar tiempo si es continuo, y sin omitir si hay interrupciones entre medio.

**🧱 Relación con vistas anteriores**

* Hereda los datos ya corregidos desde vista\_ConCubo\_2025\_Eventos.
* Se utiliza como base para **resúmenes diarios por bloque o por ID**, o como insumo para la vista final de análisis.

Perfecto Marcelo. Esta es tu **vista final integrada**, y resume todo el trabajo que hiciste. Es la que probablemente vas a usar directamente en Power BI para análisis avanzados y dashboards. Acá va la explicación clara, profesional y técnica para el README o para tu reunión con Medoro:

**✅ Vista 4: vista\_ConCubo\_2025\_Resumen\_Final**

Esta vista consolida **todas las transformaciones previas** del proyecto Medoro 2025 en un único dataset listo para análisis. Integra la secuencia validada de eventos de producción y preparación, y además vincula cada orden (ID\_Limpio) con su correspondiente entrada en la tabla externa TablaVinculadaUNION, sumando información clave como el campo saccod1 (sacabocados).

**🔧 Transformaciones clave aplicadas**

**🧩 Herencia completa de vista\_ConCubo\_2025\_Eventos\_Diario**

Incluye todos los campos calculados y corregidos previamente:

* Inicio\_Corregido, Fin\_Corregido, Fecha
* Horas separadas por tipo de estado
* Producción buena y mala
* Secuencias:
  + Nro\_Secuencia: orden cronológico por OT y máquina
  + FlagPreparacion: detección de nuevos bloques
  + SecuenciaPreparacion: numeración acumulativa de bloques reales

**🔗 Vinculación externa con TablaVinculadaUNION**

* Se realiza un LEFT JOIN para incorporar datos adicionales de producción:
* LEFT JOIN TablaVinculadaUNION VU
* ON ISNUMERIC(VU.OP) = 1
* AND TRY\_CAST(VU.OP AS INT) = s.ID\_Limpio
* Se incluye el campo saccod1 que representa el **tipo de sacabocados** o herramienta utilizada.
* La validación ISNUMERIC() evita errores de conversión si OP contiene texto (lo mismo que resolviste en ID\_Limpio).

**📊 ¿Para qué se usa esta vista?**

Es la **vista principal para análisis y visualización final**, con múltiples propósitos:

* Crear **tablas y gráficos en Power BI** con tiempos por estado, por OT, por máquina o por herramienta (saccod1).
* Filtrar eventos **por secuencia de preparación** real (bloque individual).
* Calcular KPIs:
  + Tiempo promedio por tipo de evento.
  + % de horas de preparación vs producción.
  + Eficiencia por máquina o herramienta.
* Incorporar análisis cruzado con otras tablas externas como stock, órdenes, diseños, etc.

**💡 ¿Qué resuelve esta vista?**

* Une **todo el modelado SQL previo en un solo punto de análisis**.
* Elimina la necesidad de múltiples relaciones en Power BI.
* Integra variables clave de fábrica (como saccod1) para **dimensionar eficiencia o comportamiento de cada configuración**.

**🧱 Relación con el modelo completo**

| **Vista** | **Función principal** |
| --- | --- |
| vista\_ConCubo\_2025 | Corrección inicial de fechas y tiempos |
| vista\_ConCubo\_2025\_Eventos | Estructura plana por evento |
| vista\_ConCubo\_2025\_Eventos\_Diario | Detección y secuencia de bloques reales |
| vista\_ConCubo\_2025\_Resumen\_Final | Dataset final con vinculación externa |

PARA AGREGAR AÑOS Y EXTENDER LÍMITE:

¡Exactamente, Marcelo! 💡  
Para **extender el análisis a varios años** (por ejemplo desde 2013 hasta la actualidad sin fecha de corte), solo necesitás **modificar las dos primeras vistas**:

**✅ 1. Modificar vista\_ConCubo\_2025**

Actualmente tiene este filtro:

WHERE

TRY\_CAST(Inicio AS DATETIME) >= '2025-01-01'

AND TRY\_CAST(Inicio AS DATETIME) < '2026-01-01'

🔁 **Reemplazalo por esto para que funcione desde 2013 sin tope:**

WHERE

TRY\_CAST(Inicio AS DATETIME) >= '2013-01-01'

AND ISNUMERIC(SUBSTRING(ID, PATINDEX('%[0-9]%', ID), LEN(ID))) = 1

✅ Eso incluye todos los años desde 2013 en adelante (y permitirá seguir sumando años sin tocar la vista de nuevo).

**✅ 2. Modificar vista\_ConCubo\_2025\_Eventos**

Esta tiene este filtro:

WHERE YEAR(Inicio\_Corregido) = 2025

🔁 Cambialo por:

-- Opción A: eliminar el filtro por año directamente

-- si no te interesa limitarlo:

-- sin filtro

-- Opción B: filtrar a partir de 2013

WHERE YEAR(Inicio\_Corregido) >= 2013

**🧱 Las otras vistas no necesitan ser modificadas**

Porque trabajan **sobre los datos que reciben** de las dos primeras. Si las bases (vista\_ConCubo\_2025 y vista\_ConCubo\_2025\_Eventos) ahora incluyen muchos años, todo se actualiza automáticamente.

**🧠 BONUS: agregar columna Ano (opcional)**

Si querés facilitar el análisis por año en Power BI, podrías **agregar una columna** directamente en vista\_ConCubo\_2025, así:

YEAR(DATEADD(DAY, -2, TRY\_CAST(Inicio AS DATETIME))) AS Ano

Y lo mismo en vista\_ConCubo\_2025\_Eventos:

YEAR(Inicio\_Corregido) AS Ano

Esto te permite **agrupar fácilmente en Power BI por año** sin depender de Fecha.

PASOS A REALIZAR:

¡Sí Marcelo! Eso es exactamente lo que deberían hacer. Te lo resumo paso a paso, **bien claro para explicarlo en la reunión o para enviar por mail**:

**✅ Pasos que debe seguir el equipo de la fábrica**

**🧱 1. Ingresar al SQL Server (SSMS)**

* Usar SQL Server Management Studio (SSMS).
* Conectarse al servidor correspondiente (ej. SQL\PRODUCCION o el que usen).
* Asegurarse de tener permisos para **crear vistas**.

**🧰 2. Crear las vistas en el orden correcto**

**Orden obligatorio (por dependencias):**

1. vista\_ConCubo\_2025  
   *(corrección de fechas, tiempos por estado, ID limpio, etc.)*
2. vista\_ConCubo\_2025\_Eventos  
   *(estructura de eventos por fila, usando la vista anterior)*
3. vista\_ConCubo\_2025\_Eventos\_Diario  
   *(agrega lógica de secuencia para detectar bloques de preparación)*
4. vista\_ConCubo\_2025\_Resumen\_Final  
   *(consolida todo y vincula con TablaVinculadaUNION)*

📌 **Importante**: si van a analizar desde 2013 en adelante, ya deben tener las versiones modificadas sin límite de año (te lo puedo dar ya listo).

**🔗 3. Conectar vista\_ConCubo\_2025\_Resumen\_Final a Power BI**

En Power BI Desktop:

1. Ir a **"Obtener datos" > "SQL Server"**
2. Ingresar:
   * **Servidor**: el nombre del servidor de la fábrica
   * **Base de datos**: donde crearon las vistas
   * Elegir la opción **"Importar"** (no DirectQuery)
3. Seleccionar **solo la vista vista\_ConCubo\_2025\_Resumen\_Final**
4. Cargar

⚠️ *Opcional:* pueden cargar también las otras vistas si desean tener más control o detalles, pero no es necesario.

**♻️ 4. Actualizar los datos a futuro**

* Si se cargan nuevas OTs en las tablas base (ConCubo, TablaVinculadaUNION), **las vistas se actualizan solas**.
* Solo deben actualizar Power BI haciendo clic en **"Actualizar"**.
* No necesitan tocar el código SQL si el formato de datos no cambia.

¿Querés que te lo dé en formato .txt o .md para enviarles como instructivo? También puedo darte un texto de mail para explicarles todo.