Informes y Consultas sobre un Data Mart

ASIGNATURA: ALMACENES Y MINERÍA DE DATOS

PRÁCTICA 3

CURSO: 2025/26

Pascual Albericio, Irene (NIP 871627)

Vázquez Martín, Lucía (NIP 871886)

Lucía Vázquez Martín, 871886
Irene Pascual Albericio, 871627

Índice

Lucía Vázquez Martín, 871886
Irene Pascual Albericio, 871627

1. Introducción

Esta práctica corresponde a la fase analítica del proyecto de data warehousing sobre el rendimiento de vuelos comerciales en Estados Unidos. Tras haber diseñado, implementado y poblado el data mart en las prácticas anteriores, el objetivo ahora es explotar la información almacenada mediante consultas SQL avanzadas y la elaboración de informes visuales que faciliten la toma de decisiones.

Se trabajará sobre una base de datos analítica (OLAP) con datos reales de vuelos de marzo de 2025, diseñando consultas que incluyan agregaciones básicas y funcionalidades OLAP como *CUBE*, *ROLLUP* y *GROUPING SETS*. Finalmente, se crearán informes mediante herramientas de BI como Pentaho o Tableau para representar y analizar los resultados obtenidos.

2. Consultas vuelos comerciales sin pasajeros

2.1. Consultas básicas

Consulta 1 (C1.1): Obtener el retraso medio de los vuelos que salen de cada aeropuerto en función de la ciudad destino, junto con el retraso medio total por aeropuerto origen (con independencia del destino).

```SQL
SELECT
    t1.aeropuerto_origen,
    t1.ciudad_destino,
    t1.retraso_medio_minutos,
    t2.retraso_total_aeropuerto
FROM (
    -- Retraso medio por ruta (aeropuerto origen -> ciudad destino)
    SELECT
        ao.id_aeropuerto,
        ao.codigo_IATA AS codigo_iata_origen,
        ao.nombre_aeropuerto AS aeropuerto_origen,
        ao.ciudad AS ciudad_origen,
        ad.ciudad AS ciudad_destino,
        ROUND(AVG(v.retardo_minutos), 2) AS retraso_medio_minutos
    FROM VUELO v
        INNER JOIN AEROPUERTO ao ON v.id_aeropuerto_origen = ao.id_aeropuerto
        INNER JOIN AEROPUERTO ad ON v.id_aeropuerto_destino = ad.id_aeropuerto
    GROUP BY ao.id_aeropuerto, ao.codigo_IATA, ao.nombre_aeropuerto, ao.ciudad, ad.id_aeropuerto,
ad.ciudad
) t1
INNER JOIN (
    -- Retraso medio total por aeropuerto origen (independiente del destino)
    SELECT
        ao.id_aeropuerto,
        ROUND(AVG(v.retardo_minutos), 2) AS retraso_total_aeropuerto
    FROM VUELO v
        INNER JOIN AEROPUERTO ao ON v.id_aeropuerto_origen = ao.id_aeropuerto
    GROUP BY ao.id_aeropuerto
) t2 ON t1.id_aeropuerto = t2.id_aeropuerto
ORDER BY t1.aeropuerto_origen, t1.ciudad_destino;
```

Lucía Vázquez Martín, 871886
Irene Pascual Albericio, 871627

Aeropuerto Origen	Ciudad Destino	Retraso Ruta (min)	Retraso Aeropuerto (min)
Abilene Regional	Dallas/Fort Worth, TX	0.00	0.00
Albuquerque International Sunport	Denver, CO	4.00	2.67
Albuquerque International Sunport	Seattle, WA	0.00	2.67
Arcata	Denver, CO	0.00	2.00
Arcata	San Francisco, CA (Metropolitan Area)	3.00	2.00
Bush Field	Charlotte, NC	0.00	0.00
Bush Field	Washington, DC (Metropolitan Area)	0.00	0.00
Albany International	Washington, DC (Metropolitan Area)	3.50	3.50
Amarillo International	Denver, CO	8.00	8.00
Anchorage International	Las Vegas, NV	6.00	6.25
Anchorage International	Seattle, WA	6.33	6.25
Aspen Pitkin County Sardy Field	Denver, CO	3.58	2.84
Aspen Pitkin County Sardy Field	Los Angeles, CA (Metropolitan Area)	1.50	2.84
Aspen Pitkin County Sardy Field	San Francisco, CA (Metropolitan Area)	0.00	2.84
Atlanta Municipal	Austin, TX	1.33	3.98
Atlanta Municipal	Boston, MA (Metropolitan Area)	0.00	3.98
Atlanta Municipal	Boston, MA (Metropolitan Area)	2.00	3.98
Atlanta Municipal	Columbus, OH	0.50	3.98
Atlanta Municipal	Dallas/Fort Worth, TX	14.50	3.98
Atlanta Municipal	Denver, CO	3.40	3.98
Atlanta Municipal	Detroit, MI	0.00	3.98
Atlanta Municipal	Houston, TX	12.50	3.98
Atlanta Municipal	Las Vegas, NV	6.00	3.98
Atlanta Municipal	New York City, NY (Metropolitan Area)	0.00	3.98
Atlanta Municipal	New York City, NY (Metropolitan Area)	5.00	3.98
Atlanta Municipal	Norfolk, VA (Metropolitan Area)	0.00	3.98
Atlanta Municipal	Orlando, FL	0.00	3.98
Atlanta Municipal	Philadelphia, PA	0.00	3.98
Atlanta Municipal	Phoenix, AZ	0.00	3.98
Atlanta Municipal	Raleigh/Durham, NC	0.00	3.98
Atlanta Municipal	Salt Lake City, UT	8.00	3.98
Atlanta Municipal	San Antonio, TX	0.00	3.98
Atlanta Municipal	San Diego, CA	11.50	3.98

Atlanta Municipal	San Francisco, CA (Metropolitan Area)	0.00	3.98
Atlanta Municipal	San Juan, Puerto Rico	8.50	3.98
Atlanta Municipal	Seattle, WA	0.00	3.98
Atlanta Municipal	Washington, DC (Metropolitan Area)	3.33	3.98
Atlanta Municipal	West Palm Beach/Palm Beach, FL	5.00	3.98
Outagamie County Regional	Chicago, IL	0.00	0.00
Outagamie County Regional	Dallas/Fort Worth, TX	0.00	0.00
Austin - Bergstrom International	Atlanta, GA (Metropolitan Area)	2.33	7.67
Austin - Bergstrom International	Chicago, IL	13.50	7.67
Austin - Bergstrom International	Denver, CO	13.00	7.67
Austin - Bergstrom International	Portland, OR	0.00	7.67
Austin - Bergstrom International	San Francisco, CA (Metropolitan Area)	11.00	7.67
... (801 aeropuertos de origen más)			

Este script SQL está diseñado para generar un informe detallado que compara la puntualidad de las rutas de vuelo frente al rendimiento general de sus aeropuertos de origen.

El núcleo del script es una consulta principal que se apoya en dos subconsultas anidadas, t1 y t2, para obtener dos métricas de retraso diferentes. La subconsulta t1 calcula el retraso medio específico para cada ruta individual, agrupando los vuelos por aeropuerto de origen y ciudad de destino. Paralelamente, la subconsulta t2 calcula el retraso medio general de cada aeropuerto de origen, promediando todos sus vuelos salientes independientemente del destino.

Finalmente, la consulta principal utiliza un **INNER JOIN** para unificar estos dos cálculos, conectando t1 y t2 a través del *id_aeropuerto* común. Esta consulta muestra para cada ruta su retraso medio, *retraso_medio_minutos*, junto al retraso medio general de su aeropuerto de origen, r*etraso_total_aeropuerto*, permitiendo una comparación directa. El informe se presenta ordenado alfabéticamente por aeropuerto de origen y ciudad de destino para facilitar su lectura y análisis.

Consulta 2 (C1.2): Obtener el retraso medio de los vuelos que salen de cada aeropuerto en función de la ciudad destino, junto con el retraso medio total por aeropuerto origen (con independencia del destino) y el retraso medio total por destino (con independencia del aeropuerto origen).

```SQL
SELECT
    t1.aeropuerto_origen,
    t1.ciudad_destino,
    t1.retraso_medio_ruta,
    t2.retraso_total_aeropuerto,
    t3.retraso_total_destino
```

Lucía Vázquez Martín, 871886
Irene Pascual Albericio, 871627

```
FROM (
    -- Retraso medio por ruta (aeropuerto origen -> ciudad destino)
    SELECT
        ao.id_aeropuerto AS id_aeropuerto_origen,
        ao.codigo_IATA AS codigo_iata_origen,
        ao.nombre_aeropuerto AS aeropuerto_origen,
        ao.ciudad AS ciudad_origen,
        ad.id_aeropuerto AS id_aeropuerto_destino,
        ad.ciudad AS ciudad_destino,
        ROUND(AVG(v.retardo_minutos), 2) AS retraso_medio_ruta
    FROM VUELO v
        INNER JOIN AEROPUERTO ao ON v.id_aeropuerto_origen = ao.id_aeropuerto
        INNER JOIN AEROPUERTO ad ON v.id_aeropuerto_destino = ad.id_aeropuerto
     GROUP BY ao.id_aeropuerto, ao.codigo_IATA, ao.nombre_aeropuerto, ao.ciudad, ad.id_aeropuerto,
ad.ciudad
) t1
INNER JOIN (
    -- Retraso medio total por aeropuerto origen (independiente del destino)
    SELECT
        ao.id_aeropuerto,
        ROUND(AVG(v.retardo_minutos), 2) AS retraso_total_aeropuerto
    FROM VUELO v
        INNER JOIN AEROPUERTO ao ON v.id_aeropuerto_origen = ao.id_aeropuerto
    GROUP BY ao.id_aeropuerto
) t2 ON t1.id_aeropuerto_origen = t2.id_aeropuerto
INNER JOIN (
    -- Retraso medio total por ciudad destino (independiente del aeropuerto origen)
    SELECT
        ad.ciudad,
        ROUND(AVG(v.retardo_minutos), 2) AS retraso_total_destino
    FROM VUELO v
        INNER JOIN AEROPUERTO ad ON v.id_aeropuerto_destino = ad.id_aeropuerto
    GROUP BY ad.ciudad
) t3 ON t1.ciudad_destino = t3.ciudad
ORDER BY t1.aeropuerto_origen, t1.ciudad_destino;
```

Aeropuerto Origen	Ciudad Destino	Retraso Ruta (min)	Retraso Aeropuerto Origen (min)	Retraso Ciudad Destino (min)
Abilene Regional	Dallas/Fort Worth, TX	0.00	0.00	4.60
Albuquerque International Sunport	Denver, CO	4.00	2.67	3.50
Albuquerque International Sunport	Seattle, WA	0.00	2.67	2.69
Arcata	Denver, CO	0.00	2.00	3.50
Arcata	San Francisco, CA (Metropolitan Area)	3.00	2.00	3.15
Bush Field	Charlotte, NC	0.00	0.00	2.12
Bush Field	Washington, DC (Metropolitan Area)	0.00	0.00	6.27
Albany International	Washington, DC (Metropolitan Area)	3.50	3.50	6.27
Amarillo International	Denver, CO	8.00	8.00	3.50
Anchorage International	Las Vegas, NV	6.00	6.25	2.13
Anchorage International	Seattle, WA	6.33	6.25	2.69
Aspen Pitkin County Sardy Field	Denver, CO	3.58	2.84	3.50

Aspen Pitkin County Sardy Field	Los Angeles, CA (Metropolitan Area)	1.50	2.84	2.84
Aspen Pitkin County Sardy Field	San Francisco, CA (Metropolitan Area)	0.00	2.84	3.15
Atlanta Municipal	Austin, TX	1.33	3.98	1.44
Atlanta Municipal	Boston, MA (Metropolitan Area)	0.00	3.98	4.55
Atlanta Municipal	Boston, MA (Metropolitan Area)	2.00	3.98	4.55
Atlanta Municipal	Columbus, OH	0.50	3.98	4.06
Atlanta Municipal	Dallas/Fort Worth, TX	14.50	3.98	4.60
Atlanta Municipal	Denver, CO	3.40	3.98	3.50
Atlanta Municipal	Detroit, MI	0.00	3.98	5.06
Atlanta Municipal	Houston, TX	12.50	3.98	11.65
Atlanta Municipal	Las Vegas, NV	6.00	3.98	2.13
Atlanta Municipal	New York City, NY (Metropolitan Area)	0.00	3.98	5.11
Atlanta Municipal	New York City, NY (Metropolitan Area)	5.00	3.98	5.11
Atlanta Municipal	Norfolk, VA (Metropolitan Area)	0.00	3.98	0.00
Atlanta Municipal	Orlando, FL	0.00	3.98	3.31
Atlanta Municipal	Philadelphia, PA	0.00	3.98	0.60
Atlanta Municipal	Phoenix, AZ	0.00	3.98	2.70
Atlanta Municipal	Raleigh/Durham, NC	0.00	3.98	2.64
Atlanta Municipal	Salt Lake City, UT	8.00	3.98	3.75
Atlanta Municipal	San Antonio, TX	0.00	3.98	0.09
Atlanta Municipal	San Diego, CA	11.50	3.98	2.82
Atlanta Municipal	San Francisco, CA (Metropolitan Area)	0.00	3.98	3.15
Atlanta Municipal	San Juan, Puerto Rico	8.50	3.98	4.25
Atlanta Municipal	Seattle, WA	0.00	3.98	2.69
Atlanta Municipal	Washington, DC (Metropolitan Area)	3.33	3.98	6.27
Atlanta Municipal	West Palm Beach/Palm Beach, FL	5.00	3.98	2.78
Outagamie County Regional	Chicago, IL	0.00	0.00	4.75
Outagamie County Regional	Dallas/Fort Worth, TX	0.00	0.00	4.60
Austin - Bergstrom International	Atlanta, GA (Metropolitan Area)	2.33	7.67	2.89
Austin - Bergstrom International	Chicago, IL	13.50	7.67	4.75
Austin - Bergstrom International	Denver, CO	13.00	7.67	3.50
Austin - Bergstrom International	Portland, OR	0.00	7.67	3.00
Austin - Bergstrom International	San Francisco, CA (Metropolitan Area)	11.00	7.67	3.15
... (801 aeropuertos de origen más)				

Esta segunda consulta añade a la primera la subconsulta t3 y un INNER JOIN adicional para integrar.

La nueva subconsulta t3 calcula el retraso medio total de todos los vuelos que llegan a una ciudad de destino específica, independientemente del aeropuerto de origen, agrupando los vuelos por *ad.ciudad*.

Posteriormente, el nuevo **INNER JOIN** conecta este cálculo con la consulta principal. Esto permite que el informe final muestre el retraso de la ruta, *retraso_medio_ruta*, no solo junto a la media de su aeropuerto de origen (t2), sino también junto a la media general de su ciudad de destino (t3), ofreciendo una comparativa mucho más completa.

2.2. Consultas propuestas (operadores de GROUP BY)

Consulta 1 (Análisis por temporada y fin de semana) (C2.1): Analiza la puntualidad según la época del año y si es fin de semana o no, usando *CUBE* para mostrar todas las combinaciones.

```sql
SELECT
    f.temporada,
    f.es_fin_semana,
    COUNT(*) as total_vuelos,
    ROUND(AVG(v.tiempo_real_minutos), 2) as duracion_promedio,
    ROUND(AVG(v.retardo_minutos), 2) as retraso_promedio,
    ROUND(SUM(v.distancia_km), 2) as distancia_total,
    ROUND(AVG(v.tiempo_real_minutos - v.tiempo_estimado_minutos), 2) as diferencia_tiempo
FROM VUELO v
JOIN FECHA f ON v.id_fecha = f.id_fecha
GROUP BY CUBE(f.temporada, f.es_fin_semana)
ORDER BY f.temporada NULLS LAST, f.es_fin_semana NULLS LAST;
```

TEMPORADA	ES_FIN_SEMANA	TOTAL_VUELOS	DURACION_PROMEDIO	RETRASO_PROMEDIO	DISTANCIA_TOTAL	DIFERENCIA_TIEMPO
Invierno	Fin de semana	315	154.33	4.18	457,652.84	-4.71
Invierno	Entre semana	713	147.52	3.63	968,151.59	-5.69
Invierno	Subtotal	1,028	149.61	3.8	1,425,804.43	-5.39
Primavera	Fin de semana	219	150.25	5.04	301,437.43	-4.15
Primavera	Entre semana	415	150.36	3.39	584,866.34	-5.54
Primavera	Subtotal	634	150.32	3.96	886,303.77	-5.06
Total por tipo	Fin de semana	534	152.66	4.53	759,090.27	-4.48
Total por tipo	Entre semana	1,128	148.57	3.55	1,553,017.93	-5.63
TOTAL GENERAL		**1,662**	**149.88**	**3.86**	**2,312,108.2**	**-5.26**

Hay más vuelos en invierno (1028) que en primavera (634), y los fines de semana tienen más retrasos (4.53 min) que entre semana (3.55 min). Los vuelos entre semana son ligeramente más rápidos que los de fin de semana.

Consulta 2 (Análisis por año, trimestre y mes) (C2.2): Muestra cómo se distribuyen los vuelos a lo largo del año, con totales automáticos por trimestre y año usando *ROLLUP*.

```sql
SELECT
    f.agno,
    f.trimestre,
    f.nombre_mes,
    COUNT(*) as num_vuelos,
    ROUND(AVG(v.retardo_minutos), 2) as retraso_promedio,
    ROUND(SUM(v.distancia_km), 2) as km_totales,
    ROUND(AVG(v.tiempo_real_minutos), 2) as duracion_promedio
FROM VUELO v
JOIN FECHA f ON v.id_fecha = f.id_fecha
GROUP BY ROLLUP(f.agno, f.trimestre, f.nombre_mes)

ORDER BY f.agno NULLS LAST, f.trimestre NULLS LAST, f.nombre_mes NULLS LAST;
```

AGNO	TRIMESTRE	NOMBRE_MES	NUM_VUELOS	RETRASO_PROMEDIO	KM_TOTALES	DURACION_PROMEDIO
2025	1	Marzo	1,662	3.86	2,312,108.2	149.88
2025	1	Subtotal Trimestre 1	1,662	3.86	2,312,108.2	149.88
2025	Subtotal Año 2025		1,662	3.86	2,312,108.2	149.88
TOTAL GENERAL			**1,662**	**3.86**	**2,312,108.2**	**149.88**

Todos los vuelos son de marzo de 2025 (1662 vuelos totales), con un retraso promedio de 3.86 minutos y una distancia total de 2.3 millones de km. El *ROLLUP* genera subtotales por año, trimestre y mes.

Consulta 3 (Análisis por ciudad de origen, flujos entre estados, totales por estado y total general) (C2.3): *GROUPING SETS* hace análisis independientes sin usar *UNION*.

```sql
SELECT
    ap_origen.estado,
    ap_origen.ciudad,
    ap_destino.estado as estado_destino,
    COUNT(*) as total_vuelos,
    ROUND(AVG(v.retardo_minutos), 2) as retraso_promedio,
    ROUND(AVG(v.distancia_km), 2) as distancia_promedio,
    ROUND(SUM(v.distancia_km), 2) as distancia_total,
    GROUPING(ap_origen.estado) as grp_estado_origen,
    GROUPING(ap_origen.ciudad) as grp_ciudad_origen,
    GROUPING(ap_destino.estado) as grp_estado_destino
FROM VUELO v
JOIN AEROPUERTO ap_origen ON v.id_aeropuerto_origen = ap_origen.id_aeropuerto
JOIN AEROPUERTO ap_destino ON v.id_aeropuerto_destino = ap_destino.id_aeropuerto
GROUP BY GROUPING SETS (
    (ap_origen.estado, ap_origen.ciudad),  -- Vuelos por ciudad origen
    (ap_origen.estado, ap_destino.estado), -- Flujo entre estados
    (ap_origen.estado),                     -- Total por estado origen
    ()                                      -- Total general
)
ORDER BY grp_estado_origen, grp_ciudad_origen, grp_estado_destino;
```

Lucía Vázquez Martín, 871886
Irene Pascual Albericio, 871627

Estado Origen	Ciudad Origen	Estado Destino	Total Vuelos	Retraso Prom (min)	Dist Prom (km)	Dist Total (km)	GRP Est Orig	GRP Ciud Orig	GRP Est Dest
Virginia	Richmond, VA	—	1	10.00	469.93	469.93	0	0	1
Florida	Sarasota/Bradenton, FL	—	3	18.67	1,306.78	3,920.35	0	0	1
Illinois	Peoria, IL	—	1	0.00	1,081.48	1,081.48	0	0	1
Virginia	Washington, DC (Metropolitan Area)	—	87	4.98	818.28	71,190.76	0	0	1
New York	New York City, NY (Metropolitan Area)	—	107	4.65	2,169.99	232,189.14	0	0	1
Illinois	Chicago, IL	—	115	2.70	812.80	93,472.08	0	0	1
California	Los Angeles, CA (Metropolitan Area)	—	91	2.74	2,482.47	225,904.67	0	0	1
California	San Francisco, CA (Metropolitan Area)	—	81	3.25	1,863.44	150,938.39	0	0	1
Colorado	Denver, CO	—	121	2.24	1,060.58	128,330.38	0	0	1
Georgia	—	Virginia	9	2.33	855.63	7,700.69	0	1	0
California	—	New York	37	6.24	4,020.74	148,767.39	0	1	0
California	—	Hawaii	30	3.77	4,016.86	120,505.77	0	1	0
Hawaii	—	California	29	3.10	3,981.40	115,460.49	0	1	0
Colorado	—	Colorado	64	2.58	210.80	13,491.10	0	1	0
Hawaii	—	Hawaii	48	0.48	226.38	10,866.26	0	1	0
Illinois	—	—	117	2.76	817.65	95,665.61	0	1	1
California	—	—	217	2.93	1,992.69	432,413.56	0	1	1
Colorado	—	—	171	2.53	908.82	155,407.53	0	1	1
Florida	—	—	122	7.87	1,485.79	181,266.40	0	1	1
New York	—	—	116	4.81	2,056.43	238,546.03	0	1	1
Hawaii	—	—	90	1.26	2,053.25	184,792.47	0	1	1
—	—	—	1,662	3.86	1,391.16	2,312,409.92	1	1	1
... (579 filas más)									

De los cuatro niveles de análisis geográficos generados mediante *GROUPING SETS* se obtienen diversos resultados significativos. En el nivel de ciudades, destacan Chicago con 115 vuelos y Denver con 121, siendo los principales puntos de conexión del conjunto analizado. En cuanto a los flujos estado-estado, la ruta entre California y Nueva York lidera con 37 vuelos, consolidándose como el corredor aéreo más activo. Respecto a los totales por estado de origen, California presenta el mayor número de vuelos (217), mientras que Florida registra el mayor retraso promedio con 7.87 minutos. Finalmente, en el total general se observa un retraso promedio de 3.86 minutos y una distancia total recorrida de aproximadamente 2.3 millones de kilómetros. Además, se identifican dos hallazgos relevantes: Midland/Odessa (TX) presenta retrasos particularmente elevados, con un promedio de 38.67 minutos, mientras que los vuelos interislas en Hawaii destacan por su excepcional puntualidad, con solo 0.48 minutos de retraso promedio.

Consulta 4 (Análisis de las 3 ruta más transitadas) (C2.4): Top 3 rutas más transitadas con análisis de ranking y contribución porcentual. *ROW_NUMBER*, *DENSE_RANK*, *RATIO_TO_REPORT* y *PERCENT_RANK* son window functions que calculan rankings y porcentajes sobre las filas sin colapsarlas, a diferencia de *GROUP BY* que agrupa y reduce el resultado.

```sql
SELECT *
FROM (
    SELECT
        ap_origen.ciudad || ' → ' || ap_destino.ciudad as ruta,
        ap_origen.ciudad as ciudad_origen,
        ap_destino.ciudad as ciudad_destino,
        COUNT(*) as total_vuelos,
        ROUND(AVG(v.retardo_minutos), 2) as retraso_promedio,
        ROUND(AVG(v.distancia_km), 2) as distancia_km,
        ROUND(AVG(v.tiempo_real_minutos), 2) as duracion_promedio,
        ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC) as ranking,
        DENSE_RANK() OVER (ORDER BY COUNT(*) DESC) as ranking_denso,
        ROUND(RATIO_TO_REPORT(COUNT(*)) OVER () * 100, 2) as porcentaje_trafico,
        ROUND(COUNT(*) * 100.0 / SUM(COUNT(*)) OVER (), 2) as pct_manual,
        PERCENT_RANK() OVER (ORDER BY COUNT(*)) as percentil
    FROM VUELO v
    JOIN AEROPUERTO ap_origen ON v.id_aeropuerto_origen = ap_origen.id_aeropuerto
    JOIN AEROPUERTO ap_destino ON v.id_aeropuerto_destino = ap_destino.id_aeropuerto
    GROUP BY ap_origen.ciudad, ap_destino.ciudad
)
WHERE ranking <= 3
ORDER BY ranking;
```

Ranking	Ruta	Ciudad Origen	Ciudad Destino	Total Vuelos	Retraso Prom (min)	Distancia (km)	Duración Prom (min)	% Tráfico	Percentil
1	New York City → Los Angeles	New York City, NY (Metropolitan Area)	Los Angeles, CA (Metropolitan Area)	28	3.57	3,977.08	370.75	1.68%	10000
2	Los Angeles	Los Angeles, CA	New York City, NY	27	4.19	3,976.86	317.67	1.62%	0.9987

	→ New York City	(Metropolitan Area)	(Metropolitan Area)						
3	Aspen → Denver	Aspen, CO	Denver, CO	26	3.58	201.17	72.15	1.56%	0.9974

Las dos rutas más transitadas corresponden a los vuelos bidireccionales entre Nueva York y Los Ángeles, con 28 y 27 vuelos respectivamente. En conjunto representan el 3.3% del tráfico total y, debido a los vientos, presentan diferencias notables en su duración. Por otro lado, la tercera ruta más frecuente es Aspen–Denver, con 26 vuelos y un trayecto corto de carácter turístico. Las métricas confirman que no existen empates en el número de vuelos y que cada ruta supone menos del 2% del tráfico total, lo que evidencia un mercado fragmentado. Todas ellas se ubican en el percentil superior de popularidad y presentan retrasos promedio muy similares (entre 3.57 y 4.19 minutos).

Consulta 5 (Análisis de eficiencia de aerolíneas con modelos de aviones y franjas horarias): Identifica qué combinaciones tienen mejor velocidad promedio, menor desviación de tiempos estimados y retrasos más consistentes. *STDDEV* calcula la desviación estándar para medir consistencia de retrasos, cuanto menor sea, más predecible será. Este análisis cruzado revela patrones operativos como "¿Qué modelos son más eficientes en cada franja horaria?", "¿Qué aerolíneas gestionan mejor ciertos modelos?" o ¿Qué modelo tiene mejor rendimiento en los recorridos actuales?

```sql
SELECT
    a.nombre_aerolinea,
    av.fabricante,
    av.nombre_modelo,
    h.franja_horaria,
    COUNT(*) as total_vuelos,
            ROUND(AVG(v.distancia_km / NULLIF(v.tiempo_real_minutos, 0) * 60), 2) as
velocidad_promedio_kmh,
    ROUND(AVG(v.tiempo_real_minutos - v.tiempo_estimado_minutos), 2) as desviacion_tiempo,
    ROUND(AVG(v.retardo_minutos), 2) as retraso_promedio,
    ROUND(STDDEV(v.retardo_minutos), 2) as desviacion_std_retraso,
    GROUPING(a.nombre_aerolinea) as grp_aerolinea,
    GROUPING(av.fabricante) as grp_fabricante,
    GROUPING(av.nombre_modelo) as grp_modelo,
    GROUPING(h.franja_horaria) as grp_franja
FROM VUELO v
JOIN AEROLINEA a ON v.id_aerolinea = a.id_aerolinea
JOIN AVION av ON v.id_avion = av.id_avion
JOIN HORA h ON v.id_hora = h.id_hora
GROUP BY CUBE(a.nombre_aerolinea, av.fabricante, av.nombre_modelo, h.franja_horaria)
HAVING COUNT(*) >= 1
ORDER BY grp_aerolinea, grp_fabricante, grp_modelo, grp_franja, total_vuelos DESC;
```

Aerolínea	Fabricante	Modelo	Franja Horaria	Vuelos	Vel (km/h)	Desv Tiempo	Retraso	Desv Std	A	F	M	H
Republic Airline	Embraer	Embraer ERJ170-200LR	12:00-19:59	84	440.85	-2.36	5.46	10.69	0	0	0	0

SkyWest Airlines Inc.	Embraer	Embraer ERJ170-200LR	12:00-19:59	83	344.51	-9.27	2.54	8.31	0	0	0	0
PSA Airlines Inc.	GE	GE CL-600-2C10	12:00-19:59	75	395.56	2.79	8.64	15.80	0	0	0	0
Hawaiian Airlines Inc.	Airbus	Airbus A321-271N	20:00-23:59	6	761.53	-15.33	2.67	6.53	0	0	0	0
Republic Airline	Embraer	Embraer ERJ170-200LR	— Todas —	168	425.82	-2.48	4.91	10.10	0	0	0	1
PSA Airlines Inc.	GE	GE CL-600-2C10	— Todas —	153	395.20	-0.74	5.95	12.86	0	0	0	1
Republic Airline	Embraer	— Todos —	12:00-19:59	84	440.85	-2.36	5.46	10.69	0	0	1	0
Republic Airline	Embraer	— Todos —	— Todas —	168	425.82	-2.48	4.91	10.10	0	0	1	1
Republic Airline	— Todos —	Embraer ERJ170-200LR	12:00-19:59	84	440.85	-2.36	5.46	10.69	0	1	0	0
— Todas —	Embraer	Embraer ERJ170-200LR	12:00-19:59	226	412.41	-3.93	5.22	16.15	1	0	0	0
— Todas —	— Todos —	— Todos —	12:00-19:59	776	476.28	-4.58	4.63	12.66	1	1	1	0
— Todas —	— Todos —	— Todos —	05:00-11:59	715	497.17	-5.67	3.23	8.36	1	1	1	0
— Todas —	— Todos —	— Todos —	20:00-23:59	163	536.99	-6.44	3.07	7.28	1	1	1	0
— TOTAL GENERAL —	—	—	—	1,662	492.30	-5.26	3.86	10.52	1	1	1	1
... (399 filas más)												

En los 16 niveles de agregación generados mediante *CUBE* sobre aerolínea, fabricante, modelo y franja horaria, se observa que Republic Airline con el modelo Embraer ERJ170-200LR lidera con 84 vuelos, mientras que Delta Air Lines y Hawaiian Airlines registran las mayores velocidades en horarios nocturnos. La mayoría de los vuelos operan en horario diurno, y Embraer domina en volumen, aunque Airbus alcanza la mayor velocidad promedio. En conjunto, los resultados muestran llegadas ligeramente adelantadas.

2.3. Sobre la sentencia CREATE DIMENSION y similares

Para este apartado se utilizó la base de datos **Oracle Database 21c Express Edition (XE)**. El objetivo era evaluar la relevancia, utilidad y necesidad de la sentencia *CREATE DIMENSION* y el parámetro *STAR_TRANSFORMATION_ENABLED* para el data mart de vuelos desarrollado.

El primer paso fue definir las 6 dimensiones del modelo (*DIM_FECHA*, *DIM_AEROPUERTO*, *DIM_AVION*, *DIM_HORA*, *DIM_AEROLINEA*, *DIM_OPERADORA*) mediante la sintaxis *CREATE DIMENSION.*

Antes de crear las dimensiones, se necesitan gestionar los privilegios, ya que el usuario utilizado (vuelos_user) no disponía de estos permisos por defecto. Para ello, es necesario conectarse como *SYSDBA* a *XEPDB1* para otorgar todos los privilegios necesarios:

Shell
```
docker exec -it oracle-xe bash
sqlplus / as sysdba
```

SQL
```
ALTER SESSION SET CONTAINER = XEPDB1;
-- Otorgar privilegio para crear dimensiones
GRANT CREATE DIMENSION TO vuelos_user;
-- Otorgar privilegio para eliminar dimensiones
GRANT DROP ANY DIMENSION TO vuelos_user;
-- Otorgar privilegio para modificar dimensiones
GRANT ALTER ANY DIMENSION TO vuelos_user;
-- ----------------------------------------------------------------------------
-- Crear el rol
CREATE ROLE plustrace;
-- Vista de sesiones activas
GRANT SELECT ON v_$session TO plustrace;
-- Vista de estadísticas por sesión
GRANT SELECT ON v_$sesstat TO plustrace;
-- Vista de nombres de métricas
GRANT SELECT ON v_$statname TO plustrace;
-- Otorgar rol plustrace a vuelos_user
GRANT plustrace TO vuelos_user;
```

Con los permisos concedidos, se crean por separado las 6 dimensiones, definiendo niveles, jerarquías y atributos. A continuación, se muestran todas juntas.

SQL
```
-- 1. DIMENSIÓN FECHA
CREATE DIMENSION dim_fecha
LEVEL fecha IS FECHA.id_fecha
LEVEL mes IS FECHA.mes
LEVEL trimestre IS FECHA.trimestre
LEVEL año IS FECHA.agno
HIERARCHY jerarquia_temporal (
    fecha CHILD OF mes CHILD OF trimestre CHILD OF año
)
ATTRIBUTE fecha DETERMINES (
    FECHA.fecha_completa,
    FECHA.dia,
    FECHA.nombre_dia,
    FECHA.nombre_mes,
    FECHA.es_fin_semana,
    FECHA.temporada
```

```
);
-- 2. DIMENSIÓN AEROPUERTO
CREATE DIMENSION dim_aeropuerto
LEVEL aeropuerto IS AEROPUERTO.id_aeropuerto
LEVEL ciudad IS AEROPUERTO.ciudad
LEVEL estado IS AEROPUERTO.estado
LEVEL pais IS AEROPUERTO.pais
HIERARCHY jerarquia_geografica (
    aeropuerto CHILD OF ciudad CHILD OF estado CHILD OF pais
)
ATTRIBUTE aeropuerto DETERMINES (
    AEROPUERTO.codigo_IATA,
    AEROPUERTO.nombre_aeropuerto,
    AEROPUERTO.latitud,
    AEROPUERTO.longitud,
    AEROPUERTO.num_habitantes,
    AEROPUERTO.zona_horaria,
    AEROPUERTO.codigo_estado
);
-- 3. DIMENSIÓN AEROLINEA
CREATE DIMENSION dim_aerolinea
LEVEL aerolinea IS AEROLINEA.id_aerolinea
ATTRIBUTE aerolinea DETERMINES (
    AEROLINEA.codigo_DOT,
    AEROLINEA.nombre_aerolinea,
    AEROLINEA.esta_activo
);
-- 4. DIMENSIÓN OPERADORA
CREATE DIMENSION dim_operadora
LEVEL operadora IS OPERADORA.id_operadora
ATTRIBUTE operadora DETERMINES (
    OPERADORA.codigo_IATA,
    OPERADORA.nombre_operadora,
    OPERADORA.esta_activo
);
-- 5. DIMENSIÓN AVION
CREATE DIMENSION dim_avion
LEVEL avion IS AVION.id_avion
LEVEL modelo IS AVION.nombre_modelo
LEVEL fabricante IS AVION.fabricante
HIERARCHY jerarquia_aviones (
    avion CHILD OF modelo CHILD OF fabricante
)
ATTRIBUTE avion DETERMINES (
    AVION.matricula,
    AVION.codigo_modelo,
    AVION.capacidad_maxima,
    AVION.agno_fabricacion,
    AVION.esta_activo
);
-- 6. DIMENSIÓN HORA
CREATE DIMENSION dim_hora
LEVEL hora IS HORA.id_hora
LEVEL franja IS HORA.franja_horaria
HIERARCHY jerarquia_temporal_hora (
    hora CHILD OF franja
)
ATTRIBUTE hora DETERMINES (
    HORA.hora_completa,
    HORA.hora,
```

```
    HORA.minuto,
    HORA.segundo,
    HORA.es_hora_punta
);
```

La siguiente consulta a *USER_DIMENSIONS* confirmó que todas las dimensiones fueron compiladas y se encuentran en estado *valid*.

```SQL
SELECT dimension_name, compile_state, invalid
FROM USER_DIMENSIONS
ORDER BY dimension_name;
```

La principal utilidad de *CREATE_DIMENSION* es informar sobre las jerarquías al optimizador para que pueda aplicar las optimizaciones avanzadas, siendo la principal la **Star Transformation**. Esta optimización requiere además de las dimensiones, el uso de **índices BITMAP** en las claves foráneas de la tabla de hechos.

Se encontró una limitación importante en Oracle XE, ya que, al intentar crear los índices bitmap, saltó el error *ORA-02158: invalid CREATE INDEX option.* Esto muestra que el Oracle utilizado (Express Edition), tiene un soporte limitado para esta funcionalidad. En su defecto, creamos índices *B-TREE estándar.*

```SQL
CREATE INDEX idx_vuelo_fecha ON VUELO(id_fecha);
CREATE INDEX idx_vuelo_hora ON VUELO(id_hora);
CREATE INDEX idx_vuelo_avion ON VUELO(id_avion);
CREATE INDEX idx_vuelo_aerolinea ON VUELO(id_aerolinea);
CREATE INDEX idx_vuelo_operadora ON VUELO(id_operadora);
CREATE INDEX idx_vuelo_aero_origen ON VUELO(id_aeropuerto_origen);
CREATE INDEX idx_vuelo_aero_destino ON VUELO(id_aeropuerto_destino);
```

Finalmente, para realizar las pruebas de *STAR_TRANSFORMATION_ENABLED* se diseñó una consulta analítica con filtros en múltiples dimensiones para evaluar el comportamiento de Star Transformation:

```SQL
SELECT
    ap_origen.estado,
    f.nombre_mes,
    COUNT(*) as total_vuelos,
    AVG(v.retardo_minutos) as retraso_promedio
FROM VUELO v
JOIN AEROPUERTO ap_origen ON v.id_aeropuerto_origen = ap_origen.id_aeropuerto
JOIN FECHA f ON v.id_fecha = f.id_fecha
WHERE ap_origen.estado IN ('California', 'Texas', 'Florida')
  AND f.trimestre = 1
GROUP BY ap_origen.estado, f.nombre_mes;
```

1. Configuración sin índices, con Star Transformation deshabilitado

```sql
ALTER SESSION SET STAR_TRANSFORMATION_ENABLED = FALSE;
-- Ejecutamos consulta --
```

Elapsed: 00:00:00.01
Plan de ejecución:
Plan hash value: 1684164639

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	108	12 (9)	00:00:01
1	HASH GROUP BY		3	108	12 (9)	00:00:01
* 2	HASH JOIN		193	6948	11 (0)	00:00:01
* 3	HASH JOIN		193	4632	8 (0)	00:00:01
* 4	TABLE ACCESS FULL	AEROPUERTO	18	252	3 (0)	00:00:01
5	TABLE ACCESS FULL	VUELO	1662	16620	5 (0)	00:00:01
* 6	TABLE ACCESS FULL	FECHA	31	372	3 (0)	00:00:01

2. Configuración sin índices, con Star Transformation habilitado

```sql
ALTER SESSION SET STAR_TRANSFORMATION_ENABLED = TRUE;
-- Ejecutamos consulta --
```

Elapsed: 00:00:00.13
Plan de ejecución:
Plan hash value: 1684164639

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	108	12 (9)	00:00:01
1	HASH GROUP BY		3	108	12 (9)	00:00:01
* 2	HASH JOIN		193	6948	11 (0)	00:00:01
* 3	HASH JOIN		193	4632	8 (0)	00:00:01
* 4	TABLE ACCESS FULL	AEROPUERTO	18	252	3 (0)	00:00:01
5	TABLE ACCESS FULL	VUELO	1662	16620	5 (0)	00:00:01
* 6	TABLE ACCESS FULL	FECHA	31	372	3 (0)	00:00:01

3. Configuración con índices, con Star Transformation habilitado

```sql
ALTER SESSION SET STAR_TRANSFORMATION_ENABLED = TRUE;
-- Ejecutamos consulta --
```

Elapsed: 00:00:00.06
Plan de ejecución:
Plan hash value: 1684164639

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	108	12 (9)	00:00:01
1	HASH GROUP BY		3	108	12 (9)	00:00:01
* 2	HASH JOIN		193	6948	11 (0)	00:00:01
* 3	HASH JOIN		193	4632	8 (0)	00:00:01
* 4	TABLE ACCESS FULL	AEROPUERTO	18	252	3 (0)	00:00:01
5	TABLE ACCESS FULL	VUELO	1662	16620	5 (0)	00:00:01
* 6	TABLE ACCESS FULL	FECHA	31	372	3 (0)	00:00:01

Como se puede observar en las tres configuraciones probadas, los planes de ejecución fueron idénticos (Plan hash value: 1684164639) en todos los casos, independientemente de la presencia o ausencia de índices o del estado del parámetro *STAR_TRANSFORMATION_ENABLED*.

Los tiempos de ejecución oscilaron entre 0.01 y 0.13 segundos, diferencias insignificantes que se pueden atribuir a la caché del sistema u otros factores como la variabilidad.

Las observaciones más importantes son:

- **No se aplicó Star Transformation:** no aparecieron operaciones características como *BITMAP CONVERSION TO ROWIDS, BITMAP AND* o *HASH JOIN SEMI.*
- **No se utilizaron los índices:** todas las tablas se accedieron mediante *TABLE ACCESS FULL*, ignorando los índices creados.
- **Planes idénticos:** el optimizador eligió la misma estrategia en todos los escenarios (*HASH GROUP BY → HASH JOIN → TABLE ACCESS FULL*), ya que un *TABLE ACCESS FULL* es más rápido y barato que un plan complejo de transformación.

Star Transformation no se aplicó debido a estas razones principales:

1. El dataset de 1662 registros es demasiado pequeño para esta optimización diseñada para millones de filas.
2. Las limitaciones técnicas de Oracle Express Edition en funcionalidades de data warehousing empresarial.
3. El optimizador de Oracle determinó la estrategia debido a que era la más eficiente para aplicar la transformación.

Para el caso diseñado en la práctica, *CREATE DIMENSION* no mejoró el rendimiento ya que los planes de ejecución fueron idénticos en todos los escenarios. En entornos empresariales con muchos más registros, *CREATE DIMENSION* es muy útil y necesaria, sobre todo para: documentación ejecutable de jerarquías, metadatos estructurados para el optimizador y como fundamento para funcionalidades avanzadas.

No obstante, como limitaciones, requiere privilegios especiales, es específico de Oracle o simplemente no aporta beneficios de rendimiento en datasets pequeños, como es el caso, añadiendo así complejidad.

A diferencia de Oracle, otros SGBD como PostgreSQL no disponen de una sentencia *CREATE DIMENSION* nativa. En su lugar, confían en la creación de esquemas en estrella, el uso de índices y un planificador de consultas que optimice los *star joins* de forma eficiente.

2.4. Consultas MDX

En este apartado opcional se proponía definir y probar al menos 3 consultas MDX (Multidimensional Expressions). A diferencia de SQL, MDX es un lenguaje que está diseñado para consultar cubos OLAP, permitiendo un análisis multidimensional (como slice, drill-down,…) de forma más natural.

Para ejecutar MDX, la práctica necesita un motor OLAP específico, como Pentaho BI Server/Mondrian. Se dedicó un tiempo considerable a intentar configurar este entorno, sin embargo, no se logró dicho objetivo. A continuación, se comentan los dos intentos más relevantes que se llevaron a cabo para completar esta tarea:

- **Intento 1 (Docker):** se intentaron utilizar varias imágenes de Docker para Pentaho, pero se descubrió que los repositorios de estas imágenes en Docker Hub habían sido eliminadas o privatizadas, devolviendo el error: *ERROR: pull access denied… repository does not exist or may require 'docker login'.* Para todas ellas se verificó que efectivamente sí que se estaba logueado y ese no era el motivo del fallo.

- **Intento 2 (Instalación local):** se intentó la instalación manual de Pentaho en la máquina local, pero también fue un fracaso debido a que los enlaces de descarga oficiales en SourceForge habían sido eliminados (*404 Not Found*).

Debido a estas barreras de infraestructura, no fue posible desplegar un entorno de pruebas funcional. A pesar de esto, para demostrar la comprensión del lenguaje y su aplicación sobre el data mart, se definieron las 3 consultas MDX conceptuales que se habrían implementado.

Antes de formular las consultas, primero se debe haber definido un fichero XML para el esquema Mondrian, donde se define un cubo virtual (*CuboVuelos*) sobre las tablas relacionales, especificando las dimensiones, con sus jerarquías y métricas.

- ***Tabla de Hechos:*** *VUELO*

- ***Dimensiones:*** *[Fecha] [Aeropuerto Origen] [Aeropuerto Destino] [Avion]*

- ***Medidas:*** *[Measures].[Total Vuelos], [Measures].[Retraso Promedio], [Measures].[Distancia Total]*

CONSULTA 1: Total de vuelos y retraso promedio por estado y mes

```SQL
SELECT
  -- ON COLUMS especifica las métricas que queremos ver
  {[Measures].[Total Vuelos], [Measures].[Retraso Promedio]} ON COLUMNS,
  -- ON ROWS define el eje de las filas
  -- NON EMPTY elimina filas donde no hay datos
  -- Crossjoin obtiene producto cartesiano de todos los estados y los meses
  NON EMPTY Crossjoin(
    [Aeropuerto Origen].[Estado].Members,
    [Fecha].[Mes].Members
  ) ON ROWS

FROM [CuboVuelos]
-- WHERE es el filtro
WHERE ([Fecha].[Año].&[2025])
```

CONSULTA 2: Top 5 rutas con mayor número total de vuelos

```SQL
SELECT
  {[Measures].[Total Vuelos]} ON COLUMNS,
-- TopCount toma un conjunto de datos (todas las combinaciones de rutas con Crossjoin), un número
(5) y la métrica por la que ordenar
TopCount(
    Crossjoin(
      [Aeropuerto Origen].[Ciudad].Members,
      [Aeropuerto Destino].[Ciudad].Members
    ),
    5,
    [Measures].[Total Vuelos]
  ) ON ROWS
FROM [CuboVuelos]
```

CONSULTA 3: Análisis jerárquico de los vuelos del primer trimestre desde California

```SQL
-- WITH MEMBER crea una medida calculada al vuelo (Retraso Total (Horas))
WITH
  MEMBER [Measures].[Retraso Total (Horas)] AS
   ([Measures].[Retraso Promedio] * [Measures].[Total Vuelos]) / 60,
   FORMAT_STRING = "#,##0.0 'h'"
SELECT
  {[Measures].[Total Vuelos], [Measures].[Retraso Promedio], [Measures].[Retraso Total (Horas)]}
ON COLUMNS,

  -- DrilldownLevel es una función jerárquica que coge "California" y    expande hacia abajo,
  mostrando sus "hijos" (ciudades)
  -- Hierarchize asegura que la salida mantenga un orden jerárquico
  Hierarchize(
    DrilldownLevel(
      {[Aeropuerto Origen].[Estado].&[California]},
      [Aeropuerto Origen].[Ciudad]
```

```
    )
  ) ON ROWS
FROM [CuboVuelos]
WHERE ([Fecha].[Trimestre].&[1])
```

La principal conclusión de este apartado es que el mayor coste de MDX no es la complejidad del lenguaje, sino la complejidad de infraestructura necesaria. Para un data mart como el que se está utilizando actualmente (pequeño) es más práctico, flexible y eficiente hacer uso de SQL con operadores analíticos, en vez de emplear MDX.

3. Informes

3.1. Dashboard 1: Centro de control de retrasos

Este primer dashboard se centra en responder a las preguntas más críticas sobre la eficiencia operativa del data mart de vuelos, ¿dónde y cuándo se producen los peores retrasos? Está diseñado para identificar de un vistazo los principales aeropuertos problemáticos y su contexto temporal, mes y si es o no fin de semana.

El dashboard se compone de 5 elementos visuales basados en las consultas:

1. **Tarjeta "Número de vuelos totales":** muestra el valor de *TOTAL_VUELOS* obtenida de la consulta C2.1.

2. **Tarjeta "Retraso promedio general":** muestra el *RETRASO_PROMEDIO* de la consulta C2.1 filtrando por la fila *TOTAL GENERAL.*

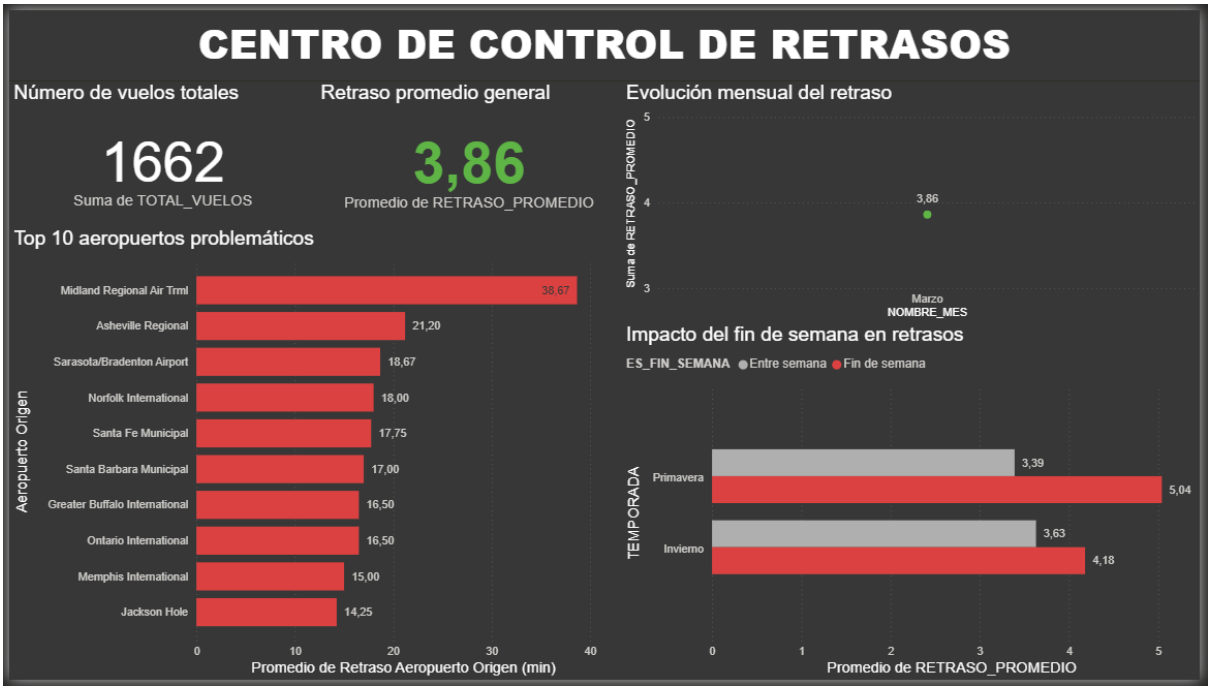
3. **Gráfico de barras apiladas "Top 10 aeropuertos problemáticos":** se hizo uso de la consulta C1.1, donde se colocó *Retraso Aeropuerto Origen (min)* en el eje X y *Aeropuerto de Origen* en el eje Y. En este caso, fue necesario cambiar la agregación de suma a promedio, ya que había aeropuertos que aparecían en múltiples rutas. Además, en *Aeropuerto de Origen* se modificó el filtrado básico por Top N para mostrar solamente los 10 aeropuertos de origen con mayores retrasos.

4. **Gráfico de barras agrupadas "Impacto del fin de semana en retrasos":** en la consulta C2.1 se excluyeron los subtotales y los totales por tipo de los campos *TEMPORADA* y *ES_FIN_SEMANA* para evitar la distorsión de los datos y mostrar fielmente la influencia de la fecha en los retrasos de los vuelos.

5. **Gráfico de líneas "Evolución mensual del retraso":** igual que en el caso anterior, pero utilizando la consulta C2.2, se filtró únicamente por los datos de *NOMBRE_MES* que realmente eran meses de año (actualmente solo está marzo). Por ello, el gráfico muestra un solo punto. Este gráfico está diseñado para funcionar con las futuras cargas incrementales del data mart. A medida que se añadan los datos de meses posteriores (abril, mayo, junio, etc.), este gráfico se poblará automáticamente, permitiendo a los analistas y gestores detectar tendencias,

estacionalidad (¿empeoran los retrasos en verano?) y el impacto de las medidas de mejora a lo largo del tiempo.

Para el diseño se ha empleado un tema oscuro y colores de alerta, tonalidades de rojo, para mostrar la información de forma limpia a la vez que se muestra una historia visual que permite percibir de manera instantánea los problemas que tiene actualmente el data mart. Los elementos se han colocado de una manera estratégica para que la vista pase de algo genérico como es la información del total de vuelos mensuales y su retraso promedio a cosas más específicas como los retrasos de cada uno de los aeropuertos y si estos retrasos se producen más en fin de semana o no.



3.2. Dashboard 2: Análisis de rendimiento de rutas

A diferencia del dashboard 1, que se centra en el dónde y cuándo, este dashboard responde a las preguntas ¿qué aeropuertos y cómo producen los mayores retardos? Su objetivo es analizar el rendimiento de rutas específicas, identificando cuáles son las que tienen mayor tráfico y cómo se comportan en términos de retrasos.

El dashboard se compone de 4 elementos visuales basados en las consultas:

1. **Treemap “Rutas por volumen de vuelos”:** se usa la consulta C2.4, pero aplicada a las 50 rutas más frecuentadas, en vez de únicamente a las tres primeras. Cada rectángulo tiene un tamaño definido por *Total Vuelos* y en función de su *Retraso Prom (min)* se muestra dicho rectángulo en verde, amarillo o rojo, gracias al formato degradado dentro de los controles avanzados de los colores.

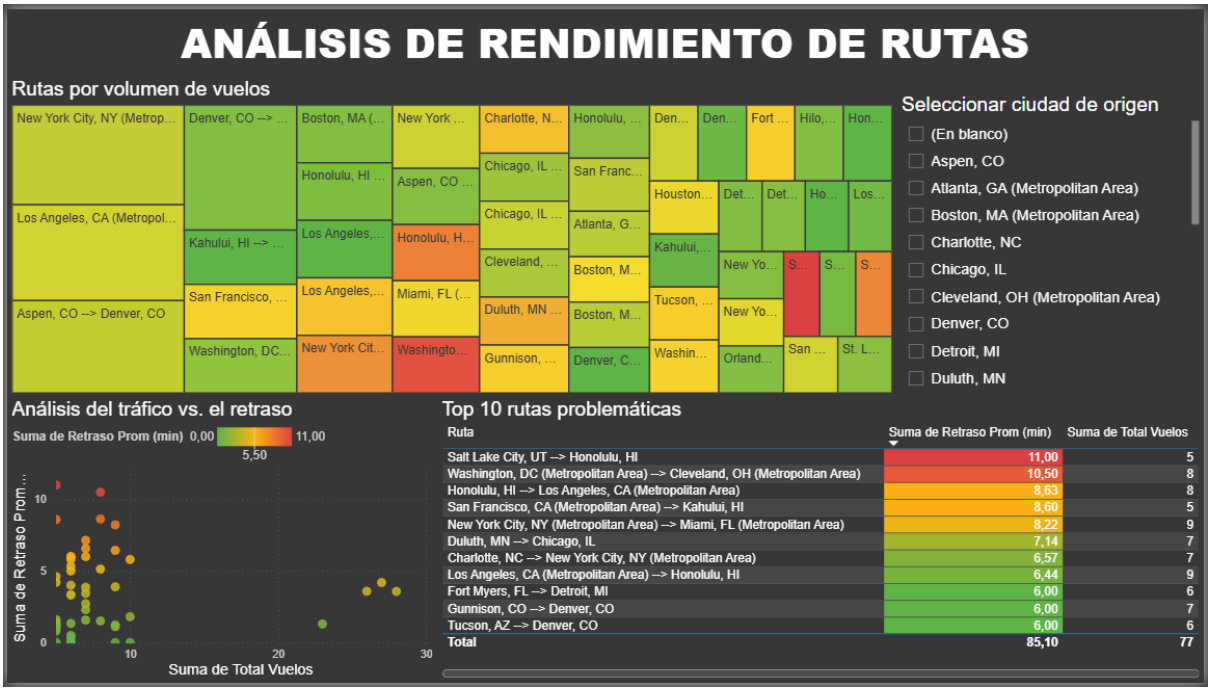
2. **Gráfico de dispersión “Análisis del tráfico vs. el retraso”:** este gráfico es el más analítico, ya que utilizando la misma consulta que antes, se coloca *Total Vuelos* en el eje X, *Retraso Prom (min)* en el eje Y y *Ruta* en detalles. Esto permite localizar

problemas críticos que se muestran en la gráfica en la zona superior derecha, ya que es donde se encuentran las rutas con alto tráfico y mayor retraso, frente a las rutas ideales que serían las localizadas en la zona inferior derecha debido a su alto tráfico y bajo retraso. A su vez, también es necesario prestar atención a los puntos que se encuentran en la zona superior izquierda porque reflejan rutas con poco tráfico, pero mucho retraso y podrían corresponder a problemas ocultos.

3. **Tabla "Top 10 rutas problemáticas":** dadas las 50 rutas más frecuentadas de la consulta utilizada hasta ahora se usa Top N del mismo modo que en el dashboard 1 para filtrarlas en función de su *Retraso Prom (min)* ordenado de mayor a menor. Esto permite mostrar una lista de acción directa frente a la gráfica de dispersión anterior, y además, responde a la pregunta de qué rutas son las que tienen un mayor retraso independientemente del tráfico que tengan.

4. **Segmentación de datos "Filtro de aeropuerto":** utilizando la misma consulta, se coge *Ciudad Origen* para mostrar un comportamiento interactivo que permite al usuario filtrar todo el dashboard con un solo clic para ver instantáneamente solo el rendimiento de *sus* rutas, haciendo el informe relevante y accionable a nivel local.

Se mantiene la misma estética que en el dashboard 1, y en este caso, la disposición de elementos permite observar en primer lugar cuáles son las rutas más grandes, posteriormente, cómo se comportan en tema de rendimiento y finalmente, cuáles son las peores.



3.3. Conclusiones sobre los informes

Los dos dashboards muestran visualmente la información del data mart de vuelos, centrándose concretamente en la eficiencia operativa y el rendimiento de las rutas.

El primer dashboard se centra en responder a las preguntas críticas de dónde y cuándo se producen los peores resultados. Para ello, identifica los 10 aeropuertos más problemáticos y analiza el impacto que tiene en los retrasos el hecho de que sea fin de semana o no.

Respecto al segundo dashboard, este responde a preguntas como qué aeropuertos, qué rutas,… Su objetivo es analizar el rendimiento de rutas específicas mediante un gráfico de dispersión para localizar problemas críticos, las rutas ideales y otros problemas, tanto con tráfico y retraso alto como bajo. Todo esto se complementa con una tabla de las top 10 rutas problemáticas para una acción directa y un filtro interactivo por ciudad de origen.

En la práctica, estos informes permiten identificar dónde enfocar los esfuerzos de mejora. El primer dashboard señala claramente qué 10 aeropuertos tienen el peor rendimiento y destaca que los fines de semana son un punto débil en la operativa. Con esta información, la gerencia puede asignar recursos, como personal o equipos de soporte, de manera más precisa en lugar de aplicar medidas generales.

El segundo informe, por su parte, ayuda a priorizar las intervenciones. El gráfico de dispersión distingue entre las rutas críticas (las que tienen mucho tráfico y muchos retrasos) y otros problemas. Esto es fundamental para decidir qué rutas deben revisarse primero para tener un mayor impacto. Además, el filtro por ciudad resulta muy funcional, ya que permite a un responsable local analizar el rendimiento de sus rutas específicas sin perderse en el conjunto global de datos.

4. Distribución del trabajo

En el desarrollo de esta práctica, todas las tareas se llevaron a cabo de forma conjunta por los integrantes del equipo. La colaboración fue continua en todas las fases del diseño de consultas y la generación de informes, combinando esfuerzos en el análisis de las funcionalidades SQL, el diseño e implementación de las consultas, la configuración de las herramientas de BI y la elaboración de la memoria.

Las actividades desarrolladas conjuntamente incluyen:

- **Formulación de las consultas SQL básicas,** definiendo y probando las sentencias requeridas para obtener los resultados esperados.
- **Diseño y prueba de las consultas analíticas avanzadas,** investigando y desarrollando las 5 consultas adicionales no triviales que utilizan operadores OLAP como *CUBE*, *ROLLUP* y *GROUPING SETS*, además de otros no vistos en clase tales cual *top-n* y *window functions*.
- **Análisis de funcionalidades específicas del SGBD,** investigando la relevancia y utilidad de la sentencia *CREATE DIMENSION* de Oracle y el parámetro *STAR_TRANSFORMATION_ENABLED,* realizando las pruebas pertinentes en Oracle.
- **Investigación y análisis del apartado opcional sobre MDX**, incluyendo los intentos de configuración del entorno y la definición conceptual de las 3 consultas MDX.

- **Selección y configuración de herramientas de BI,** evaluando y utilizando la herramienta *Power BI* para conectar con la base de datos, definir las visualizaciones y crear los cuadros de mando solicitados.
- **Redacción de los apartados correspondientes en la memoria**, integrando el código SQL, las capturas de los informes, el análisis de las funcionalidades del SGBD y las conclusiones.

En conjunto, el proyecto fue resultado de una colaboración equitativa y coordinada entre todos los miembros del grupo, sin asignaciones individuales diferenciadas.

Apartados del trabajo realizados	Lucía	Irene
Consultas básicas	2	2
Consultas propuestas (operadores de GROUP BY)	2	2
Análisis de funcionalidades específicas del SGBD	0	1.5
Investigación MDX	0	1.5
Informes	7	3.5
Documentación	4	3
	15	13.5

5. Conclusiones

En esta práctica se ha completado el ciclo del proyecto de *data warehousing*, partiendo de la construcción del *data mart*, que se había llevado a cabo en sesiones anteriores, a su explotación analítica. Esto ha ayudado a comprender la importancia de un modelo dimensional bien diseñado y cómo esto facilita la extracción de conocimiento.

La aplicación de los operadores OLAP de SQL, como *CUBE*, *ROLLUP* y *GROUPING SETS,* ha permitido comprobar en la práctica su enorme potencia para generar agregaciones multidimensionales y sumarios complejos con una única sentencia, lo cual hubiese requerido múltiples consultas y uniones en el caso de usar *GROUP BY*. La exploración de otras funciones analíticas, como las *top-n* y *window functions*, ha profundizado la comprensión de las capacidades de SQL para el análisis.

La investigación sobre funcionalidades específicas del SGBD, como la sentencia *CREATE DIMENSION* y el parámetro *STAR_TRANSFORMATION_ENABLED* en Oracle, ha proporcionado una visión valiosa sobre los mecanismos internos que utilizan los gestores de bases de datos para optimizar las consultas en esquemas en estrella, aunque su uso en la práctica no sea siempre directo.

Adicionalmente, la investigación del apartado opcional de MDX, a pesar de las barreras que impidieron la ejecución práctica, ha demostrado que el principal desafío de MDX no es la

complejidad de su lenguaje sino la sobrecarga de infraestructura necesaria para su despliegue.

Finalmente, el uso de una herramienta de BI como *Power BI* ha sido clave para tangibilizar el valor de los datos. Este paso ha permitido conectar los resultados de las consultas SQL con un entorno visual e interactivo, demostrando que la visualización es un componente esencial para interpretar patrones, comunicar hallazgos y facilitar la toma de decisiones.