☑ 1. Preguntas y respuestas sobre Data Warehousing & Arquitectura

1. ¿Cuál es la diferencia entre Data Lake, Data Warehouse y Lakehouse?

- **Data Lake:** almacena datos crudos (no estructurados, semiestructurados y estructurados). Bajo costo, ideal para ML, pero requiere gobernanza para evitar el "data swamp".
- **Data Warehouse:** datos estructurados, alta calidad, optimizado para analítica (OLAP). Modelos dimensionales.
- **Lakehouse:** combina ambos. Ej: Delta Lake, BigQuery, Snowflake (con capacidades de semi-estructurado).

---

2. Explica la diferencia entre ETL y ELT.

- **ETL:** extraes → transformas → cargas. Adecuado cuando necesitas controlar la transformación antes de llegar al destino (tradicional).
- **ELT:** extraes → cargas → transformas. Moderno en DW cloud (Snowflake/BigQuery). Usa el poder del warehouse para transformar.

---

3. ¿Qué es un *staging layer* en un DW?

Zona donde se almacenan los datos crudos antes de procesarlos. Permite reproducibilidad, auditabilidad y debugging.

---

4. ¿Qué son SCD Type 1 y Type 2?

- **Type 1:** no mantiene historial. Sobrescribe.
- **Type 2:** crea registros nuevos para cambios, manteniendo versiones (con campos `valid_from`, `valid_to`, `is_current`).

☑ 2. Preguntas de Modelado Dimensional

5. ¿Qué es una tabla de hechos?

Tabla transaccional que almacena métricas (medidas cuantitativas). Ej: ventas, clics, pedidos.

---

6. ¿Cuándo usarías un modelo estrella vs copo de nieve?

- **Estrella:** más simple, mejor performance; dimensiones desnormalizadas.
- **Copo de nieve:** normalizado; útil para gobernanza o dimensiones muy grandes.

---

7. ¿Qué son las *slowly changing dimensions* y por qué importan?

Dimensiones cuyos atributos cambian lentamente con el tiempo. Importantes para análisis histórico fiel.

☑ 3. Preguntas de SQL (con respuestas)

8. Escribe una consulta para obtener los 3 productos más vendidos por país.

```
SELECT country, product_id, total_sales
FROM (
    SELECT
        country,
        product_id,
        SUM(amount) AS total_sales,
        ROW_NUMBER() OVER (PARTITION BY country ORDER BY SUM(amount)
DESC) AS rn
    FROM sales
    GROUP BY country, product_id
) t
WHERE rn <= 3;
```

---

9. ¿Cómo detectar duplicados en una tabla?

```
SELECT id, COUNT(*)
FROM users
GROUP BY id
HAVING COUNT(*) > 1;
```

---

10. Explica CROSS JOIN, INNER JOIN y LEFT JOIN.

- **INNER:** une donde hay coincidencia.
- **LEFT:** mantiene todo de la izquierda.
- **CROSS:** producto cartesiano.

☑ 4. Preguntas de Python

11. ¿Cómo manejarías errores en un pipeline?

- Excepciones controladas
- Logging estructurado
- Retries
- Validación de datos antes de procesar

---

12. ¿Qué librerías usarías para validación de datos?

- **Pydantic** para esquemas
- **Great Expectations** para tests de calidad
- **pandera** para validar estructuras de DataFrames

---

13. ¿Cómo optimizarías un proceso lento en Python?

- Vectorizar con pandas o NumPy
- Multiprocessing / threading
- Caching
- Reducir IO

☑ 5. Snowflake

14. ¿Qué es el *virtual warehouse* en Snowflake?

Es la capa de cómputo; puede escalar horizontal (multi-cluster) o verticalmente (tamaño). Storage y compute están separados.

---

15. ¿Qué es *Time Travel*?

Funcionalidad para consultar datos como estaban hace X tiempo (1–90 días). Útil para auditoría, recuperación y debugging.

---

16. ¿Qué son Streams y Tasks?

- **Streams:** capturan cambios (CDC).
- **Tasks:** programan ejecuciones de SQL o procesos.

☑ 6. dbt

17. ¿Qué es dbt?

Herramienta ELT que permite crear transformaciones versionadas, testeables, documentadas y modulares dentro del entorno SQL del warehouse.

---

18. Explica la estructura de un proyecto dbt.

- `models/` → SQL models
- `tests/` → validaciones
- `macros/`
- `snapshots/`
- `dbt_project.yml` → configuración

---

19. ¿Qué son los *tests* en dbt?

Validaciones de integridad y calidad:

- Únicos
- No nulos
- Relaciones
- Tests personalizados (Jinja)

---

20. ¿Cómo se logra transparencia y mantenibilidad en dbt?

- Modelos claros y pequeños
- Nomenclaturas (stg_, int_, dim/fct_)
- Documentación automática
- Tests
- Lineage con DAG

☑ 7. Orquestación (Airflow / Prefect / Dagster)

21. ¿Qué es un DAG en Airflow?

Un grafo dirigido acíclico que define dependencias entre tareas.

---

22. ¿Qué significa idempotencia?

Ejecutar un job varias veces sin generar efectos inconsistentes (muy importante en pipelines).

---

23. ¿Cómo manejas fallos en un DAG?

- Retries
- Alerts
- Failure callbacks
- Reprocesar sólo particiones afectadas

☑ 8. GCP

24. Explica cómo funciona BigQuery.

Motor serverless de análisis columnar:

- Separa storage/compute

- Paga por TB escaneado
- Ideal para ELT
- Ofrece particionamiento y clustering

---

25. ¿Qué rol cumple Cloud Composer?

Implementación de Airflow administrada en GCP. Se usa para orquestar pipelines.

---

26. Diferencia entre Pub/Sub y Dataflow.

- **Pub/Sub** → mensajería/eventos
- **Dataflow** → procesamiento en streaming/batch (Apache Beam)

☑ 9. Gobernanza y Metadata

27. ¿Qué es data lineage?

Rastrea el origen y el recorrido de los datos, útil para auditorías, debugging, gobernanza.

---

28. ¿Qué herramientas para metadata management conoces?

- DataHub
- Amundsen
- Collibra
- Alation

---

29. ¿Qué es un data contract?

Especificación formal del esquema y la calidad mínima que debe cumplir un dataset; evita rupturas en pipelines.

☑ 10. Seguridad

30. ¿Qué es Row-Level Security?

Controla qué filas puede ver un usuario según reglas (muy usado en BI).

---

31. ¿Qué es el enmascaramiento de datos?

Ocultar datos sensibles con reglas (masking policy). Snowflake tiene funciones nativas.

☑ 11. CI/CD para Data Engineering

32. ¿Cómo integras dbt en CI/CD?

- Cada PR ejecuta:
  - `dbt compile`
  - `dbt run` (en ambiente dev)
  - `dbt test`
- Deployment a prod después de aprobación

---

33. ¿Qué pruebas incluirías en CD?

- Tests de schema
- Tests de calidad
- Linting (sqlfluff)
- Validación de dependencias

☑ 12. BI (Power BI / Tableau / Looker)

34. ¿Por qué es importante el semantic layer?

Permite definir métricas y relaciones de forma centralizada, evitando ambigüedad.

---

35. ¿Cómo optimizas dashboards?

- Extracts
- Minimizar visuales
- Reducir joins complejos
- Evitar cálculos row-level en tiempo real