Resumen SQL/DES

1. Comandos de Sistema DES

/abolish: Elimina todas las tablas y vistas.
/multiline on: Permite consultas multilínea.
/duplicates off: Elimina duplicados automáticamente.

2. DDL - Definición de Datos

CREATE TABLE: Crea una nueva tabla.

```
create table tabla(
  col1 tipo primary key,
  col2 tipo,
  col3 tipo
);
```

Tipos: `string`, `int`, `float`, etc.
PRIMARY KEY: Define clave primaria (simple o compuesta).

3. DML - Manipulación de Datos

INSERT: Inserta tuplas en una tabla.

```
insert into tabla
  values('val1','val2',123);
insert into tabla(col1,col2)
  values('val1','val2');
```

NULL: Representa valores desconocidos/no aplicables.

4. Consultas SELECT

SELECT básico:

```
select columnas from tabla
  where condicion;
```

SELECT *: Selecciona todas las columnas.
DISTINCT: Elimina duplicados en resultados.

5. Operadores Lógicos

AND, OR, NOT: Operadores booleanos.
Comparación: =, ¡¿, ¡, ¿, ¡=, ¿=
IS NULL / IS NOT NULL: Comprueba valores nulos.
IN / NOT IN: Pertenencia a conjunto.

6. VISTAS (CREATE VIEW)

Las vistas son consultas almacenadas que pueden referenciarse como tablas.

```
create view nombre_vista as
  select ... from ... where ...;
```

Ventajas: Reutilización, abstracción, simplicidad.

7. Operaciones de Conjuntos

UNION: Unión de resultados (elimina duplicados).

```
select dni from programadores
union
select dni from analistas;
```

UNION DISTINCT: Unión eliminando duplicados explícitamente.
INTERSECT: Intersección de resultados.

```
select dni from programadores
intersect
select dni from analistas;
```

EXCEPT: Diferencia de conjuntos (A - B).

```
select dni from programadores
except
select dni from analistas;
```

EXCEPT ALL: Diferencia manteniendo duplicados.

8. JOIN - Combinación de Tablas

JOIN implícito (producto cartesiano con WHERE):

```
select * from tabla1, tabla2
  where tabla1.id = tabla2.id;
```

NATURAL JOIN: Join por columnas con mismo nombre.

```
select * from tabla1
  natural join tabla2;
```

JOIN explícito:

```
select * from tabla1
  join tabla2 on cond;
```

9. Funciones de Agregación

SUM(col): Suma de valores.
COUNT(*): Cuenta tuplas.
AVG(col): Media aritmética.
MAX(col): Valor máximo.
MIN(col): Valor mínimo.

```
select dni, sum(horas) as total
from distribucion
group by dni;
```

10. GROUP BY y HAVING

GROUP BY: Agrupa filas por columna(s).

```
select col, count(*) from tabla
  group by col;
```

HAVING: Filtra grupos (después de GROUP BY).

```
select col, count(*) from tabla
  group by col
  having count(*) > 5;
```

Diferencia: WHERE filtra antes de agrupar, HAVING después.

11. Subconsultas

Subconsulta escalar: Devuelve un único valor.

```
select * from tabla where col =
  (select max(col) from tabla);
```

Subconsulta en IN:

```
where dni in
  (select dni from analistas);
```

Subconsulta en FROM: Tabla derivada.

```
select * from
  (select dni from prog
   union select dni from anal);
```

12. DIVISION

Operador específico de DES para encontrar elementos que se relacionan con todos los de otro conjunto.

```
select dniEmp from
(select codigoPr,dniEmp
 from distribucion)
division
(select codigoPr from dist
 where dniEmp='5');
```

Interpretación: Empleados asignados a todos los proyectos del empleado '5'.

13. Operaciones Aritméticas

Se pueden realizar operaciones en SELECT:

```
select codigo, horas*1.2
  from distribucion;
```

Operadores: +, -, *, /

14. Alias y RENAME

AS: Define alias para columnas/tablas.

```
select dni as identificador,
  sum(horas) as total
from distribucion;
```

RENAME (sintaxis DES):

```
rename (dniEmp as dni)
  (distribucion);
```

15. Consultas Complejas

Combinando múltiples operaciones:

```
create view vista_compleja as
(select dni,nombre,codigoPr
 from (select * from prog
       union select * from anal)
 join dist on dni = dniEmp)
union
(select dni,nombre,codigo
 from (select * from prog
       union select * from anal)
 join proy on dni = dniDir);
```

16. Buenas Prácticas

- Usar vistas para consultas complejas reutilizables
- Aplicar filtros WHERE antes de JOIN cuando sea posible
- Usar DISTINCT solo cuando sea necesario (impacto en rendimiento)
- Nombrar columnas con alias descriptivos
- Dividir consultas complejas en vistas intermedias
- Considerar NULL en comparaciones (NULL != NULL)

17. Orden de Ejecución SQL

Orden lógico de evaluación:

1. FROM (tablas/joins)
2. WHERE (filtrado de filas)
3. GROUP BY (agrupación)
4. HAVING (filtrado de grupos)
5. SELECT (proyección)
6. UNION/INTERSECT/EXCEPT
7. ORDER BY (no usado en ejemplos)

18. Ejemplos Prácticos

Empleados sin teléfono:

```
select dni,nombre from prog
where telefono is null
union
select dni,nombre from anal
where telefono is null;
```

Suma de horas por empleado:

```
select dni, sum(horas) as total
from (select dni from prog
      union select dni from anal)
join dist on dni = dniEmp
group by dni;
```

Proyectos sin analistas:

```
select codigo from proyectos
except
select distinct codigo from
(select codigoPr as codigo,
        dniEmp from dist)
join analistas on dniEmp = dni;
```