## Agenda

- DQL - SELECT
    - LIMIT
    - ORDER BY
    - WHERE
        - Relational Operators
        - Logical Operators
        - NULL Operators
        - IN,BETWEEN,LIKE Operators
- DML - UPDATE, DELETE
- DDL - Truncate, DROP
- DELETE vs TRUNCATE vs DROP

## Limit

- to limit no of rows in output

```sql
-- show all rows from emp
SELECT * FROM emp;

-- show first 5 rows from emp
SELECT * FROM emp LIMIT 5;

-- show first 5 rows from emp
SELECT * FROM emp LIMIT 10;

-- show ename and sal of first 5 rows
SELECT ename,sal FROM emp LIMIT 5;

-- show ename and sal of first 10 rows
SELECT ename,sal FROM emp LIMIT 10;

-- show ename,sal of 5 rows after first 3 rows
SELECT ename,sal FROM emp LIMIT 3,5;
```

## Order By

- used to order the output rows in ascending or descending manner

```sql
-- display all the employees ordered on their salary in desc order
SELECT * FROM emp ORDER BY sal DESC;

-- display all the employees ordered on their deptno in asc order
SELECT * FROM emp ORDER BY deptno ASC;
SELECT * FROM emp ORDER BY deptno; -- default -> ASC
```

```
-- display all the employees ordered on their deptno in asc and jobs in desc order
SELECT * FROM emp ORDER BY deptno ASC ,job DESC;

-- display all the employees ordered on their deptno in asc, jobs in desc and
ename in asc order
SELECT * FROM emp ORDER BY deptno ASC ,job DESC,ename ASC;
```

## Limit & OrderBY

```
-- display top 3 employees as per salary
SELECT * FROM emp ORDER BY sal DESC LIMIT 3;

-- display emp whose name is last alphabetically
SELECT * FROM emp ORDER BY ename DESC LIMIT 1;

-- display emp with lowest sal.
SELECT * FROM emp ORDER BY sal LIMIT 1;

-- display emp with third lowest salary
SELECT * FROM emp ORDER BY sal ASC LIMIT 2,1;

-- display emp with second highest salary
SELECT * FROM emp ORDER BY sal DESC LIMIT 1,1;
```

## Orderby and computed columns

```
-- display emps ordered on the da(sal*0.5) in ASC order
SELECT ename,sal,sal*0.5 AS da FROM emp ORDER BY sal*0.5;
-- order by expression

SELECT ename,sal,sal*0.5 AS da FROM emp ORDER BY da;
-- order by column alias

SELECT ename,sal,sal*0.5 AS da FROM emp ORDER BY 3;
-- order by column no in projection
```

## Where clause

```
-- display all emps from dept 30
SELECT  * FROM emp WHERE deptno = 30;

-- display all the emps with sal > 2000;
SELECT  * FROM emp WHERE sal > 2000;
```

```
-- display all ANALYST
SELECT  * FROM emp WHERE job = "ANALYST";

-- display all emps not in dept 30;
SELECT  * FROM emp WHERE deptno != 30;
SELECT  * FROM emp WHERE deptno <> 30;

-- display all emps in sal range of 1000 to 2000
-- sal >=1000 && sal<=2000
SELECT  * FROM emp WHERE sal >= 1000 AND sal<=2000;

-- display all the analaysts and manager
-- job = "ANALYST" || job = "MANAGER"
SELECT  * FROM emp WHERE job = "ANALYST" OR job = "MANAGER";

-- display all emps who are not salesman
SELECT  * FROM emp WHERE job != "SALESMAN";
SELECT  * FROM emp WHERE job <> "SALESMAN";

-- diplay all the emps hired in 1982
SELECT  * FROM emp WHERE hire >= "1982-01-01" AND hire <= "1982-12-31";
```

## Working with NULL in WHERE clause

- NULL is used with special operators.
- relational operators cannot be used with NULL.

```

-- display all emps with comm as NULL
SELECT * FROM emp WHERE comm=NULL; -- empty set doesnt work
SELECT * FROM emp WHERE comm IS NULL;
SELECT * FROM emp WHERE comm <=> NULL;

-- display all emps with comm as NOT NULL
SELECT * FROM emp WHERE comm IS NOT NULL;
SELECT * FROM emp WHERE NOT comm IS NULL;

SELECT * FROM emp WHERE comm <> NULL; -- empty set doesnt work
```

## IN,BETWEEN and LIKE Operator

- IN Operator is similar to OR
- used to check equality for multiple values for same column
- NOT IN which is inverse of IN

```

-- display all emps working as Analyst and Manager
SELECT  * FROM emp WHERE job = "ANALYST" OR job = "MANAGER";
SELECT  * FROM emp WHERE job IN ("ANALYST","MANAGER");
```

```
-- display all emps working as Manager and emps from dept 30;
SELECT  * FROM emp WHERE  job = "MANAGER" OR deptno=30;
-- cannot use IN for different columns

-- display all emps whose sal is less than 1000 or sal is more than 3500.
SELECT * FROM emp WHERE sal <1000 OR sal >3500;
-- cannot use IN for non equality check

-- display emps whose names are JAMES, KING, MARTIN, FORD.
SELECT * FROM emp WHERE ename = "JAMES" OR ename = "KING" OR ename = "MARTIN" OR
ename = "FORD";

SELECT * FROM emp WHERE ename IN ("JAMES", "KING", "MARTIN", "FORD");

-- display all emps whose are not salesman or manager.
SELECT * FROM emp WHERE job NOT IN ("SALESMAN", "MANAGER");
SELECT * FROM emp WHERE NOT job IN ("SALESMAN", "MANAGER");
```

- Between operator is used for rang checking including both ends for same column

```
-- display all emps in sal range 1000 to 3000.
SELECT  * FROM emp WHERE sal >= 1000 AND sal<=2000;
SELECT  * FROM emp WHERE sal BETWEEN 1000 AND 2000;

-- diplay all the emps hired in 1982
SELECT  * FROM emp WHERE hire >= "1982-01-01" AND hire <= "1982-12-31";
SELECT  * FROM emp WHERE hire BETWEEN "1982-01-01" AND "1982-12-31";

-- display all managers of dept 20
SELECT * FROM emp WHERE job = 'MANAGER' AND deptno = 20;

-- display all emps whose sal is not in range 1000 to 2000.
SELECT  * FROM emp WHERE sal NOT BETWEEN 1000 AND 2000;
```

```
INSERT INTO emp(ename) VALUES ('B'),('J'),('K');
SELECT ename FROM emp ORDER BY ename;

-- display all emps whose first letter of name falls between 'B' and 'J';
SELECT ename FROM emp WHERE ename BETWEEN 'B' AND 'J';
-- JAMES & JONES are not displayed because alphabetically they come after J.

SELECT ename FROM emp WHERE ename BETWEEN 'B' AND 'K';
-- will display all names starting from B to J + will also display "K" (if
present).

SELECT ename FROM emp WHERE ename BETWEEN 'B' AND 'K' AND ename != 'K';
```

- LIKE operator is used with strings for finding similar records.
- Wildcard character is used to give pattern.
  - % : Any number of any char or Empty.
  - _ : Single occurrence of any char.
- NOT LIKE : inverse of LIKE

```sql
-- find all emps whose name start with M.
SELECT * FROM emp WHERE ename LIKE 'M%';

-- find all emps whose name ends with H.
SELECT * FROM emp WHERE ename LIKE '%H';

-- find all emps whose name contain U.
SELECT * FROM emp WHERE ename LIKE '%U%';

-- find all emps whose name contains A twice.
SELECT * FROM emp WHERE ename LIKE '%A%A%';

-- display all emps whose first letter of name falls between 'B' and 'J';
SELECT ename FROM emp WHERE ename BETWEEN 'B' AND 'J' OR ename LIKE 'J%';

-- display emps having 4 letter name.
SELECT * FROM emp WHERE ename LIKE '____';

-- display all emps whose name contains R on 3rd position.
SELECT * FROM emp WHERE ename LIKE '__R%';

-- display emps with 4 letter name and 3rd pos is R
SELECT * FROM emp WHERE ename LIKE '__R_';
```

## Practice Examples

```sql
-- display emp with highest sal in range 1000 to 2000.
SELECT * FROM emp WHERE sal BETWEEN 1000 AND 2000 ORDER BY sal DESC LIMIT 1;

-- display CLERK with min sal.
SELECT * FROM emp WHERE job = "CLERK" ORDER BY sal LIMIT 1;

-- diplay fifth lowest sal from dept 20 and 30.
SELECT sal FROM emp WHERE deptno IN (20, 30);
SELECT sal FROM emp WHERE deptno IN (20, 30) ORDER BY sal;
SELECT sal FROM emp WHERE deptno IN (20, 30) ORDER BY sal LIMIT 4,1;
SELECT DISTINCT sal FROM emp
WHERE deptno IN (20, 30)
ORDER BY sal LIMIT 4,1;
```

## UPDATE

```sql
-- change the location of dept 40 to USA
UPDATE dept SET loc = "USA" WHERE deptno=40;

-- increase the price of all C Programing books by 50;
UPDATE books SET price = price + 50 WHERE subject = 'C Programming';

-- for the emp B add the empno=1,sal=1000 and comm=NULL
UPDATE emp SET empno=1, sal=1000, comm=NULL WHERE ename = 'B';
```

## DELETE

```sql
-- delete emp with ename as B
DELETE FROM emp WHERE ename = 'B';

-- delete all rows from table
- DELETE FROM newemp;

-- delete all C Programming books
- DELETE FROM books WHERE subject = "C Programming"
```

## DDL - Truncate

- TRUNCATE is DDL query.
- TRUNCATE is to delete all rows
- cannot use WHERE clause.
- Table structure is not deleted (simiar to DELETE query).

```sql
-- delete all books
TRUNCATE TABLE books;
SELECT * FROM books;
DESCRIBE books;
```

## DDL-DROP

- DROP command can be used for dropping/deleting whole table or database.
- DROP TABLE tablename;
- DROP DATABASE dbname;

```sql
SELECT * FROM dummy;
DESCRIBE dummy;
DROP TABLE dummy;
SHOW TABLES;
DESCRIBE dummy;
```

## DELETE vs TRUNCATE vs DROP

- DELETE
  - Used to delete rows (not structure).
  - All rows or as per WHERE condition.
  - DML operation
  - DML ops can be rollbacked (undo/discard) using transaction.
  - In most RDBMS, slower operation.
- TRUNCATE
  - Used to delete rows (not structure).
  - All rows.
  - DDL operation.
  - DDL ops cannot be rollbacked.
  - In most RDBMS, faster operation.
- DROP
  - Used to delete rows as well as struct.
  - DDL operation.
  - DDL ops cannot be rollbacked.
  - This is fastest operation.