# Subquery, insert, update

## Sub-queries

- A sub-query is a query that is embedded (or nested) another inside query
- Also known as a nested query or an inner query.

```
SELECT    select_list
FROM      table
WHERE     expr operator   (SELECT select_list
                           FROM   table);
```

  - The first query in the SQL statement is known as the outer query
  - The query inside the SQL statement is known as the inner query.
  - The inner query is evaluated first and the output from this query is used as the input for the outer query.
  - The inner query is normally expressed inside parentheses.

## Sub-queries

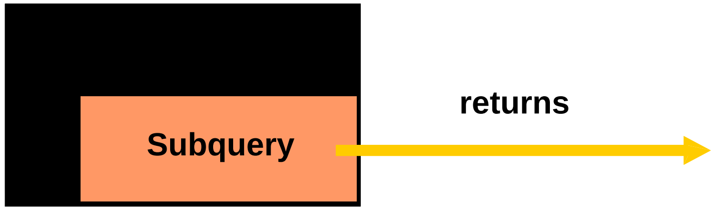
- The Oracle Server
  - executes sub-queries first
  - returns results into the clause of the main query.
- Which employee has a greater monthly salary than  Blake?

```
SELECT empno, empname, empmsal
FROM employee
WHERE empmsal > (SELECT empmsal
                 FROM employee
                 WHERE empname = 'BLAKE');
```

- Guidelines
  - Enclose sub-queries in parentheses
  - Place sub-queries on the right side of the comparison operator
  - Do not use an ORDER BY clause on a sub-query
  - Use single-row operators with single-row sub-queries
  - Use multiple-row operators with multiple-row sub-queries

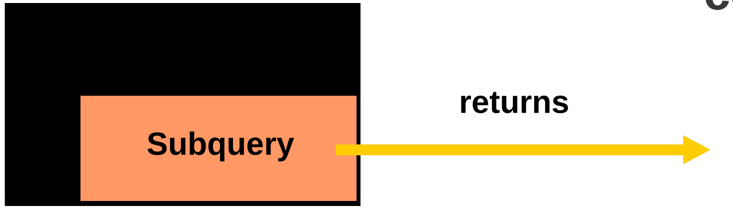
# Types of Sub-queries

Single-row sub-query (a single value)



**CLERK**

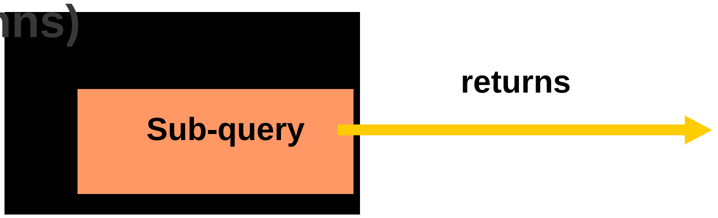
Multiple-row sub-query (a list of values – many rows, one column)



**CLERK**

**MANAGER**

Multiple-column sub-query (a virtual table – many rows, many columns)



| | |
|---|---|
| **CLERK** | **7900** |
| **MANAGER** | **7698** |

## Single-Row Sub-queries

- Display the name and job of employees who have the same job as Allen and a monthly salary greater than employee Ward

```
SELECT empname, empjob, empmsal
FROM employee
WHERE empjob = (SELECT empjob FROM employee
                       WHERE empname =
'ALLEN') AND empmsal > (SELECT empmsal
FROM employee
                     WHERE empname = 'WARD');
```

- Display the name, job and monthly salary of employees who earn the least in salary.

```
SELECT empname, empjob, empmsal
FROM employee
WHERE empmsal = (SELECT min(empmsal) FROM
employee);
```

## Single-Row Sub-queries

Which department has the most employees?

```
SELECT  d.deptno, d.deptname, count(*) FROM     department d,
employee e WHERE d.deptno = e.deptno
HAVING count(*) = (SELECT  max(count(*))
                   FROM    employee
                   GROUP BY deptno)
GROUP BY d.deptno, d.deptname;
```

Which department has the greatest average monthly salary?

```
SELECT   d.deptno, d.deptname,
avg(empmsal) FROM     department d,
employee e
WHERE d.deptno = e.deptno

HAVING avg(empmsal) =
(SELECT   max(avg(empmsal)) FROM     employee
                             GROUP BY deptno) GROUP BY
  d.deptno, d.deptname;
```

## Single-Row Sub-queries

- What is wrong with this statement?

```
SELECT empno, empname
FROM   employee
WHERE  empmsal = (SELECT MIN(empmsal)
                  FROM employee
                 GROUP BY deptno);
```

```
SQL> SELECT empno, empname
  2  FROM   employee
  3  WHERE  empmsal =  (SELECT MIN(empmsal)
  4                    FROM employee
  5                    GROUP BY deptno);
WHERE   empmsal = (SELECT MIN(empmsal)
                  *
ERROR at line 3:
ORA-01427: single-row subquery returns more than one row
```

## Multiple-Row Sub-queries

- Which employees will be displayed?

```
SELECT  empno, empname, empjob, empmsal
 FROM   employee
 WHERE  empmsal < ANY (SELECT empmsal FROM
        employee        WHERE empjob = 'SALESREP')
AND empjob <> 'SALESREP’;
```

```
SQL> SELECT empmsal FROM employee
  2  WHERE empjob = 'SALESREP';

   EMPMSAL
----------
      1600
     1250
      1250
     1500
```

## Multiple-Row Sub-queries

- Which employees will be displayed?

```
SELECT  empno, empname, empjob, empmsal
 FROM    employee
 WHERE   empmsal > ALL (SELECT AVG(empmsal) FROM employee
                                   GROUP BY deptno);
```

```
SQL> SELECT AVG(empmsal) FROM employee
  2  GROUP BY deptno;

AVG(EMPMSAL)
------------
  2916.66667
        2175
  1541.66667
```

## Multiple-Column Sub-queries

- The number of columns in the main query must match the number of columns returned from the inner query.
- Display the employees that work in the same department and have the same job as Martin.

```
SELECT     empno, empname, deptno, empjob
 FROM      employee
 WHERE      (deptno, empjob) = (SELECT deptno, empjob
                                FROM employee
                                WHERE empname =
                                'MARTIN');
```

## Relational Set Operators

- SQl data manipulation commands are set-oriented, that is they operate over entire sets of rows and columns at once. Using the set operators, you can combine two or more sets to create new sets (relations)
- Union
  - All rows selected by either query
- Union All
  - All rows selected by either query, including all duplicates
- Intersect
  - All distinct rows selected by both queries
- Minus
  - All distinct rows selected by the first query but not the second
- All set operators have equal precedence. If a SQl statement contains multiple set operators, Oracle evaluates them from the left to right if no parentheses explicitly specify another order.
- The corresponding expressions in the select lists of the component queries of a compound query must match in ***number*** and ***datatype***.

## UNION

- The UNION statement combines rows from two or more queries without including duplicate rows.
- The UNION All statement combines rows from two or more queries and retains the duplicate rows.
- The following statement combines the results with the UNION operator, which eliminates duplicate selected rows.  You must match datatypes (using the TO_CHAR, TO_DATE and TO_NUMBER functions) when columns do not exist in one or the other table:

```
SElECT 'Manager', empno, empname, empjob, mgrno
FROM employee
WHERE empno IN (SElECT mgrno FROM employee)
UNION
SElECT 'Employee', empno, empname, empjob, mgrno
FROM employee
WHERE empno NOT IN (SElECT distinct nvl(mgrno,0) FROM employee)
ORDER BY mgrno;
```

## INTERSECT

- The INTERSECT statement combines rows from two queries and returns only those rows that appear in both sets.

```
SElECT empno, empname, empjob, mgrno
FROM employee
WHERE empno IN (SElECT mgrno FROM employee)
INTERSECT
SElECT empno, empname, empjob, mgrno
FROM employee
where deptno =
20;
```

```
EMPNO                  EMPNAME  EMPJOB   MGRNO
---------------------  -------- -------- -----
--                     - JONES  -        -
7566                   SCOTT    MANAGER  7839
7788                   FORD     TRAINER  7566
7902                            TRAINER  7566

3 rows selected
```

## MINUS

- The MINUS statement combines rows from two queries and returns those rows that appear in the first set but not in the second. only

```
SElECT empno, empname, empjob, mgrno
FROM employee
MINUS
SElECT empno, empname, empjob, mgrno
FROM employee
WHERE empno IN (SElECT mgrno FROM employee);
```

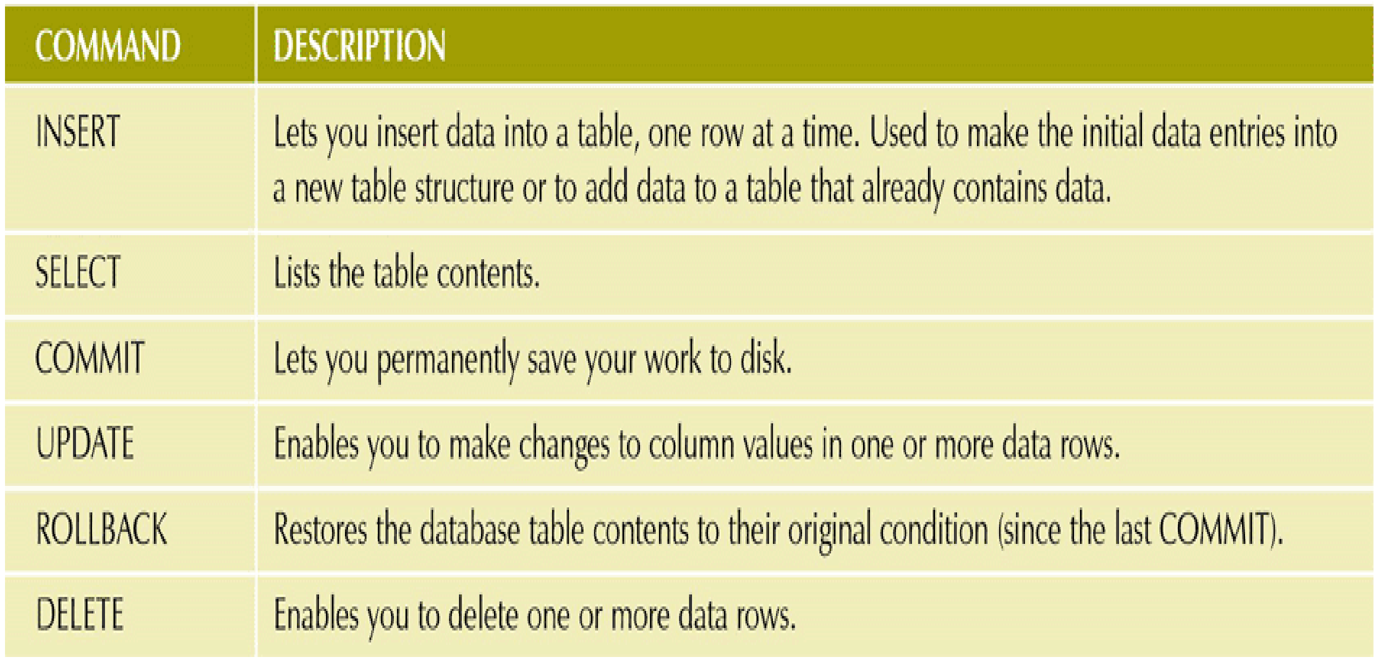
```
 EMPNO                   EMPNAME  EMPJOB   MGRNO
---------------------- -------- -------- ------
7369                   SMITH    TRAINER  7902
7499                   ALLEN    SALESREP 7698
7521                   WARD     SALESREP 7698
7654                   MARTIN   SALESREP 7698
7844                   TURNER   SALESREP 7698
7876                   ADAMS    TRAINER  7788
7900                   JONES    ADMIN    7698
7934                   MILLER   ADMIN    7782

8 rows selected
```

## Manipulating data

- There are six basic SQl data manipulation commands

TABLE 6.6 COMMON SQL DATA MANIPULATION COMMANDS



| COMMAND | DESCRIPTION |
|---|---|
| INSERT | Lets you insert data into a table, one row at a time. Used to make the initial data entries into a new table structure or to add data to a table that already contains data. |
| SELECT | Lists the table contents. |
| COMMIT | Lets you permanently save your work to disk. |
| UPDATE | Enables you to make changes to column values in one or more data rows. |
| ROLLBACK | Restores the database table contents to their original condition (since the last COMMIT). |
| DELETE | Enables you to delete one or more data rows. |

## INSERT statement

- The INSERT statement is used to enter data into a table

```
INSERT INTO          table [(column [, column...])]
VAlUES               (value [, value...]);
```

- The INSERT statement allows the insertion of data one row at a time
- If you insert a new row that contains values for each column in the table, the column list is not required in the INSERT clause
- If you do not use the column list, the values must be listed according to the default order of the columns in the table

```
INSERT INTO     department
VAlUES          (50, 'SUPPORT', 'SEATTlE',
                7788);
INSERT INTO     department (deptno, deptname, deptlocation)
VAlUES          (50, 'SUPPORT', 'SEATTlE');
```

## INSERT statement

- Can use the reserved word NUll to specify a null value for a specific column

```
INSERT INTO department
        VAlUES  (50, 'SUPPORT', 'SEATTlE', NUll);
```

- Can specify the reserved word DEFAUlT to insert the default value associated with the corresponding column. If a DEFAUlT value was not defined for the column, a NUll is inserted instead.

```
INSERT INTO employee (empno, empname, empinit, empbdate, empmsal, deptno)
        VAlUES (7999, 'DUCK', 'D.', to_date('01-JUN-1985','DD-MON-YYYY'), 0, DEFAUlT);
```

- What is the problem with this statement?

```
INSERT INTO employee (empno, empname, empinit, empbdate, empmsal, deptno)
        VAlUES (8999, 'DOO', 'S.', to_date('01-JUN-1985','DD-MON-YYYY'), 4995, 60);
```

- What is the problem with this statement?

```
INSERT INTO employee (empno, empname, empinit, empbdate, empmsal, deptno)
        VAlUES (8999, 'O'BRIEN', 'F.', to_date('01-JUN-1985', 'DD-MON-YYYY'), 4995, 40);
```

## INSERT statement

- Can use sub-queries in the VAlUES clause

```
INSERT INTO employee(empno, empname, empinit, empbdate,
empmsal)
VAlUES (1111, 'Baggins', 'B.', to_date('25-NOV-1985','DD-MON-YYYY'),
    (SElECT empmsal FROM employee WHERE empname = 'SMITH'));
```

- To insert multiple rows of data use an INSERT statement with a sub-query.

  INSERT INTO *table* [ *column* (, *column*) ]

  (*sub-query)*;

```
INSERT INTO managers(id, name, salary, bdate)
                SElECT empno, empname, empmsal, empbdate
                FROM   employee
                WHERE empjob =
                'MANAGER';
```

  – Do not use the VAlUES clause.

  – Match the number of columns in the INSERT clause to those in the sub- query.

## UPDATE statement

- The UPDATE statement allows you to change attribute values in one or more rows of a table.

  UPDATE tablename SET col1, col2, ..

  [WHERE …cond..]

| | |
|---|---|
| – UPDATE: | the table you want to update |
| – SET: | the change you want to apply |
| – WHERE: | the rows to which you want to apply the change<br>if you omit the optional WHERE clause the change is applied to all rows of the table |

## UPDATE statement

```
UPDATE  employee
SET     empjob = 'SAlESREP',
        empmsal = empmsal - 500,
        empcomm = 0,
        deptno = 30
WHERE   empno = 7876;
```

- What is the problem with this statement?

```
UPDATE  employee
SET     deptno = 60
WHERE   empno = 7876;
```

## UPDATE statement

- You can use subqueries in an UPDATE statement update rows in a to table based on values from another table.

```
UPDATE    table
SET       column = (subquery) [, column = value,
[WHERE    ...]
          condition];
```

```
UPDATE    employee
SET       empjob = (SElECT empjob FROM employee
                              WHERE empname = 'ClARK'),
          deptno = (SElECT deptno FROM employee
                      WHERE empname = 'AllEN')
WHERE     empno = 7876;
```

## DELETE statement

- The DElETE statement allows you to delete rows of data from a table.

```
DElETE FROM table
[WHERE condition]
```

  – the WHERE clause is optional, if omitted the DElETE command will delete all rows in the table

```
DElETE FROM employee;
```

```
DElETE FROM employee
WHERE empno = 7999;
```

- What is the problem with this statement?

```
DElETE FROM employee
WHERE empno = 7566;
```

## DELETE statement

- You can use subqueries in a DElETE statement to delete rows in a table based on values from another table.

```
DElETE FROM     table
[WHERE          condition = (subquery)];

DElETE FROM employee
WHERE deptno= (SElECT deptno FROM
                department WHERE deptname
                          = 'SAlES');
```

  – This query has a problem - why?