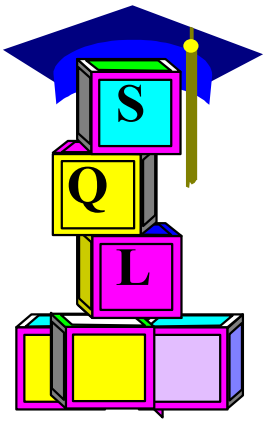
CHAPTER 2

ADVANCED STRUCTURED QUERY LANGUAGE

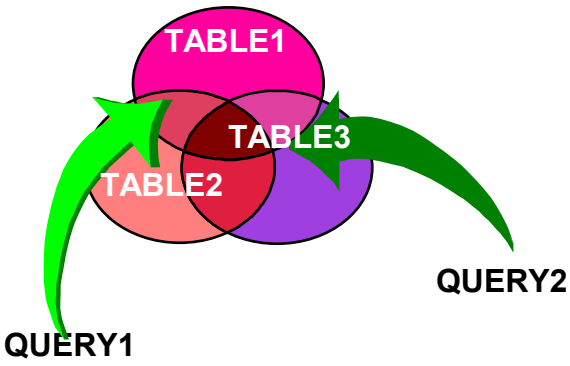


TOPIC OBJECTIVES

This section introduces the more complex SQL statements as used in ORACLE. When you finish this section, you will be able to:

- Create queries called joins which use more than one table
- Comprehend when and how to use subquery statements
- Identify and use the Union, Intersect and Difference commands
- Write SQL Select statements to not only retrieve rows but to insert, delete and update row columns

ADVANCED DML FUNCTIONS



Joins

A join allows the selection of columns from one or more tables based upon comparing two or more columns.

For example, the following figure shows a DEPT table with DEPTNO as a primary key and an EMP table with EMPNO as primary key and DEPTNO as foreign key.

DEPT TABLE

Deptno	Dname
001	SALES
002	MANUF

EMP TABLE

Empno	Mgr	Deptno	Sal	Job
10	40	001	50000	Sales
20	30	002	35800	Tech
30	..	002	43000	Mgr.
40	..	001	70500	Mgr.
50	..	003	34000	Tech

JOIN EXAMPLES

A query in which data is retrieved from more than one table. Specifically, select empno, ename, mgr and dname from the appropriate dept and emp tables

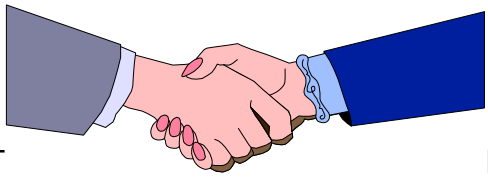
SQL:

```
SELECT EMPNO, ENAME, MGR,
       DNAME,LOC
FROM DEPT,EMP
WHERE DEPT.DEPTNO = EMP.DEPTNO
ORDER BY EMPNO
```

RESULT:

EMPNO	ENAME	MGR	DNAME
7782	CLARK	7839	ACCOUNTING
7839	KING		ACCOUNTING
7934	MILLER	7782	ACCOUNTING
7369	SMITH	7982	RESEARCH
7876	ADAMS	7788	RESEARCH
7902	FORD	7566	RESEARCH
7788	SCOTT	7566	RESEARCH
7566	JONES	7839	RESEARCH
....		.....	.....

NOTES ABOUT JOINS



DEPT EMP

In the previous example, DEPT.DEPTNO = EMP.DEPTNO is referred to as the JOIN condition

The two fields in the join condition:

- Must be either both numeric or both character
- Should be the same data type (eliminates conversion overhead)
- May have the same or different column names
- May be compared with equal (=), or with inequalities

Should be used with care.

- They can be quite demanding of the system
- One of the most powerful features of relational systems

QUALIFIED NAMES

- When the columns in the tables to be joined are the same name, you must fully qualify the columns being joined or you will have an ambiguous reference.
- Suppose the following query is needed

List all department managers and their salary

SQL:

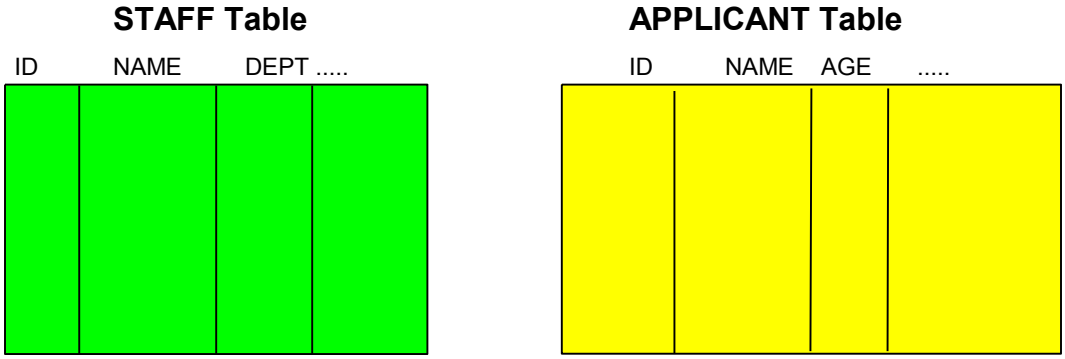
SELECT EMP.DEPTNO,ENAME,EMP.JOB,EMP.SAL,DEPT.DNAME

FROM DEPT,EMP

WHERE DEPT.DEPTNO = EMP.DEPTNO AND JOB ='MANAGER';

NOTES: Remember that different tables can contain columns with the same name

CORRELATION VARIABLES

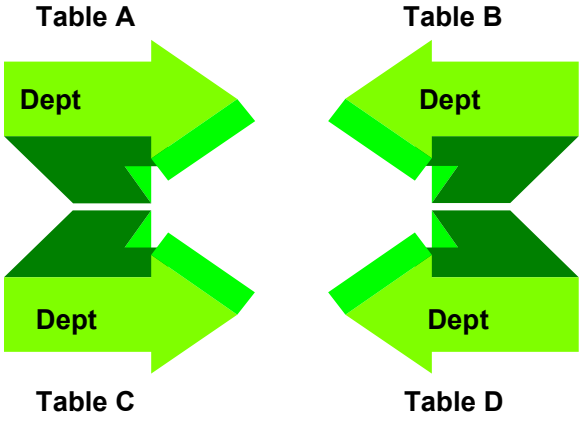


Correlation variables are aliases for table names. They are primarily used when columns in different tables have the **same** name. This occurs frequently with primary/foreign key relationships established for referential integrity.

```
SELECT A.NAME, S.NAME
FROM APPLICANT A, STAFF S
WHERE A.TEMPID=S.ID
```

- “A” and "S" are referred to as correlation variables
- Also referred to as aliases

ADVANCED DML FUNCTIONS



eEqui-join or INNER-JOIN

An equi-join is a join where a row in one table has a matching row (based upon equal values) in another table based on one (or more)column(s) in each table.

For example, if you needed to select the employee id, name, salary, and department name from the appropriate tables for employees employed by division manager 160, you would:

```
SELECT   dept, id, salary, deptname
          FROM STAFF,ORG
         WHERE  dept= deptnumb and manager = 160
          order by dept;
```

The results are shown in the following figure.

results Table

Dept	Id	salary	Deptname
10	160	22959.2	HEAD OFFICE
10	210	20020	HEAD OFFICE
10	260	21234	HEAD OFFICE
10	240	19260.3	HEAD OFFICE

ADVANCED DML FUNCTIONS

Non-equi-join

A non-equi-join uses operators other than the equal to (=) such as less than (<) and greater than (>). A non-equi-join is a join where a row in one table has a matching row (based upon unequal values) in another table based on one (or more) column(s) in each table.

Example: Find a specific employee whose salary is greater than their managers salary: Make sure you match manager 50 with employee 170 and the department is 15.

SELECT M.DEPT,S.ID, S.NAME,S.SALARY, M.NAME, M.SALARY

FROM STAFF M, STAFF S

WHERE M.JOB = 'MGR' AND M.ID = 50 AND

S.ID=170 AND s.SALARY > M.SALARY

Results:

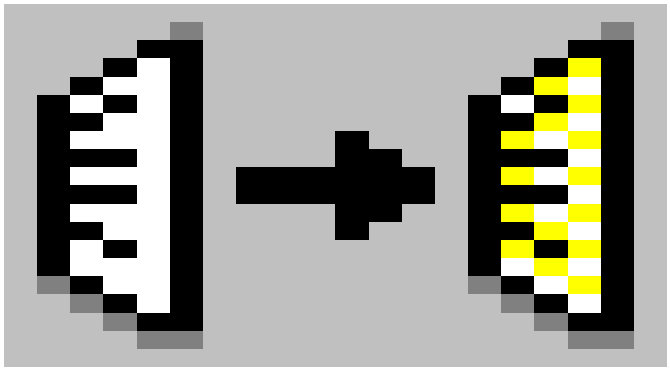
DEPT	ID	NAME	SALARY	NAME	SALARY
15	170	KERMISCH	12258.50	HANES	20659.80

(SO NO ROWS WILL BE DISPLAYED BACK BECAUSE THE EMPLOYEE HAS A SMALLER SALARY THAN THE MANAGER)

As a general rule, non-equi-joins are performed on one row from each table only)

.

ADVANCED DML FUNCTIONS



Equal Join
(Matching values)

Outer Join
(Non matching Null value

Outer Join

An outer join may be used to include rows from one table that do not have a matching row in another table.

Example: To find employees with no DEPTNO in the DEPT table use this outer join:

SELECT DEPT,ID,DEPTNUMB

FROM STAFF,ORG

WHERE DEPT = DEPTNUMB(+)

The resulting table is:

DEPT	ID	DEPTNUMB
(All matching rows and a employee with a null department value)		
NULL	80	NULL

Notice that the plus sign in parentheses (+) tells SQL this is an outer join. Put the (+) on the column on the side of the join where you expect the null value to exist.

ADVANCED DML FUNCTIONS

Do I want a value that doesn't exist in the other table

LEFT Outer Join

An outer join may be used to include rows from one table that do not have a matching row in another table.

Example: To find employees with no DEPTNO in the DEPT table use this outer join:

SELECT S.DEPT,S.ID,

Nvl(O.DEPTNAME, 'Employee has no Department')

FROM STAFF S left OUTER JOIN ORG O

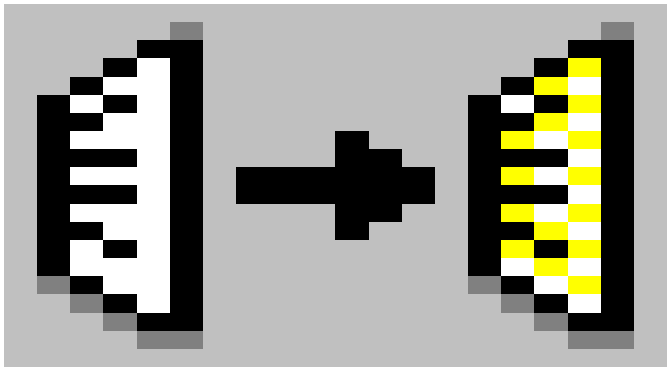
ON S.DEPT = O.DEPTNUMB.

The resulting table is:

DEPT	ID	DEPTNUMB
(All matching rows and an employee with a null department value)		
NULL	80	NULL

The missing values for DEPTNAME in the ORG table would contain nulls if the VALUE function were not utilized. The WHERE clause is replaced by the ON keyword for the outer join statement.

ANSI/ISO JOIN STATEMENTS



Equal Join
(Matching values)

Outer Join
(Non matching Null value

Natural Join or Cross Join

An natural join may be used to conform to ANSI/ISO conventions for performing an equi-join.

Example: To find employees with no DEPTNO in the DEPT table use this natural join:

SELECT S.DEPT,S.ID, **O.DEPTNAME**
FROM STAFF S NATURAL JOIN ORG O

The resulting table is printed accordingly

To perform a Cartesian product join you can use the CROSS Join statement

SELECT S.DEPT,S.ID, **O.DEPTNAME**
FROM STAFF S CROSS JOIN ORG O
ON S.DEPT = O.DEPTNUMB.

This particular query will take a 1,000 row table (like Org) and a 10,000 row table (like STAFF) and create 10,000,000 rows. This can be applicable for Financial applications where you have a table of numbers that need to be cross multiplied with another table of numbers to provide statistical numbers.

ORACLE JOIN STATEMENTS

The Inner join statement provides the ANSI standard for matching column joins. For instance:

```
Select d.deptno, empno, ename, loc
From dept d join emp e
on d.deptno = e.deptno
where salary > 20000.00
```

Since an inner join is the DEFAULT type of join you may omit the word inner

```
Select d.deptno, empno, ename, loc
From dept d inner join emp e on d.deptno = e.deptno
```

If the join clause involves columns with the same name, you may use the USING Clause, omitting the explicit join condition.

```
Select deptno, empno, ename, loc
From dept d join emp e
 USING (deptno)
```

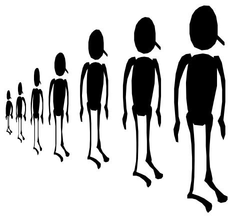
MULTIPLE TABLE JOIN

Joining More than Two tables:

You can join three or more tables with normal join syntax

```
Select oh.invoice_number, oi.product_id,
        p.description, oi.quantity
from  order_header oh JOIN order_item oi
                        on oi.invoice_number = oh.invoice_number
                        JOIN product p ON p.product_id =
                                        oi.product_id
where customer_id=1042;
Select invoice_number, product_id,
        p.description, oi.quantity
from  order_header oh JOIN order_item oi
                        using (invoice_number)
                        JOIN product p using (product_id)
where customer_id=1042;
```

More about Aliases



Aliases are very important in some advanced manipulation features like correlated subqueries.

Sometimes, it is important to have two references to the same table so that the table can be joined to itself. These aliases are called correlation variables in this case.

Task: Find all employees in each dept who have less years of experience than their manager.

SELECT S.NAME, S.DEPT,S.YEARS, M.NAME, M.DEPT, M.YEARS

FROM STAFF S, STAFF M

WHERE S.DEPT=M.DEPT AND M.JOB = 'MGR' AND S.YEARS < M.YEARS

And s.job <> 'MGR'

STAFF S

NAME	DEPT	YEARS
SANDERS	20	7
PERNAL	20	8
HANES	15	10
ROTHMAN	15	7
NGAN	15	5
KERNISCH	15	10
,,		

STAFF M

NAME	DEPT	YEARS
SANDERS	20	7
MARENGHI	38	5
HANES	15	10

RESULT:

KOONITZ	42	6	PLOTZ	42	7	
YAMAGUCHI	42	6	PLOTZ	42	7	. .
.	.	.	.	.		
.	.	.	.	.	.	.

JOIN EXERCISES: (USE THE STAFF, ORG AND APPLICANT TABLES)

1. List the department numbers and department names with their appropriate division manager's name and his/her salary. See EX71.sql

   RESULTS:

   (8 ROWS RETURNED)

```
SELECT DEPTNUMB, DEPTNAME,
            NAME, SALARY
 FROM ORG JOIN STAFF
 ON MANAGER = ID
```

2. For each manager, create a list of people (staff or other managers) in his/her department whose salary is greater than that manager's salary. See EX72.sql

   RESULTS:

   (10 ROWS RETURNED)

```
SELECT S.NAME, S.DEPT,S.salary,  M.NAME, M.DEPT, M.salary
 FROM STAFF S, STAFF M
WHERE S.DEPT=M.DEPT AND  M.JOB = 'MGR' AND S.SALARY > M.SALARY
--And s.job <> 'MGR'
```

3. For each applicant list any staff members with the same name. See EX73.sql

   RESULTS:

   (NO ROWS RETURNED)

SUBQUERIES

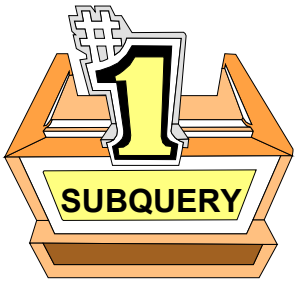
A subquery is a select-from-where expression nested within another select

```
SELECT TEMPID, NAME FROM APPLICANT
 WHERE NAME IN
    (SELECT NAME FROM STAFF
    WHERE JOB='MGR')
```

The subquery produces a value or group of values which complete the main query

- More than one level of subquery nesting can be used
- If a subquery produces no results (rows) you get an empty report

SINGLE VALUED SUBQUERIES



Provides an alternative for the need to use built in functions in the where clause..

TASK: Get the names of all applicants evaluated at the highest education level

SQL:

```
SELECT NAME FROM APPLICANT
 WHERE EDLEVEL =
(SELECT MAX(EDLEVEL)
FROM APPLICANT)
```

RESULTS:

GASPARD
JACOBS

NOTE: If "=" (Or an inequality) precedes the subquery it must be single valued

MULTIVALUED SUBQUERIES

TASK: List the id and name for each employee in the eastern division

SQL:

```
SELECT ID, NAME, 'EASTERN DIVISION' FROM STAFF
WHERE DEPT in
    (SELECT DEPTNUMB FROM ORG
    WHERE DIVISION='EASTERN')
    ORDER BY ID
```

RESULTS:

All employees from departments 15, 20, and 38

THE ANY OR ALL PARAMETER

TASK: List any employee whose salary is greater than at least one department's average salary

SQL:

```
SELECT NAME, DEPT, JOB, SALARY
 FROM STAFF
WHERE SALARY  > ANY
(SELECT AVG(SALARY) FROM STAFF
GROUP BY DEPT)
```

NOTE:
Condition is true if true for at least one value produced by the subquery

IN-LINE VIEWS

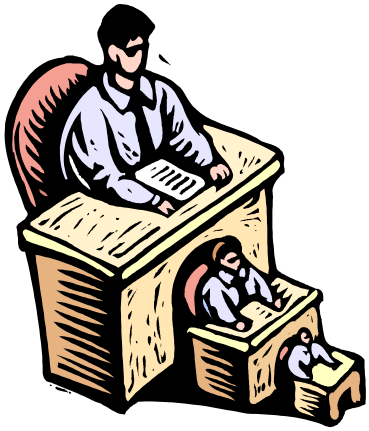


Table View View

```
SELECT DEPTNO, SUM_SALARIES
   FROM  ( SELECT DEPTNO, SUM(SAL) SUM_SALARIES
               FROM EMP
                 WHERE HIREDATE > TO_DATE('14-Jan-1980')
                   Group by deptno
                     Order by sum(sal) desc
           )
```

In Oracle you can create an in-line view (the subselect above) and it is legal to do an ORDER BV inside a view, including an in-line view as shown above.

Top-N queries are also used with inline views. It uses a virtual column called Rownum. By using an order by in a inline view you can specify only the top 10 rows to be brought back thereby assisting performance. For example:

```
SELECT ename, job, sal, rownum
FROM     (select ename, job, sal from emp
              Order by sal desc)
WHERE   rownum <=10;
```

THE EXISTS PARAMETER (CORRELATED SUBQUERY)

TASK: List each department in Org with at least one employee with more than 8 years of service

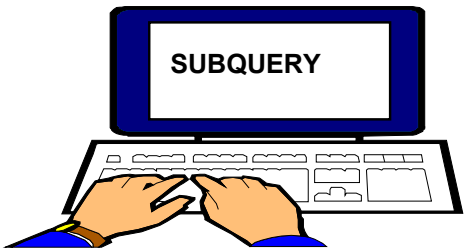
SQL:

```
SELECT DEPTNUMB FROM ORG
WHERE EXISTS
(SELECT 1 FROM STAFF
WHERE DEPT=DEPTNUMB AND YEARS > 8)
```

NOTE:

If subquery produces at least one value, the department is included in the result

EXERCISE 8: USE ORG, STAFF AND/OR APPLICANT TABLES



1 List the managers in the Midwest division. See ex81.sql

RESULTS:

NAME	DEPT
PLOTZ	42
FRAYE	51

```
SELECT NAME, DEPT, 'MIDWEST'
FROM STAFF
WHERE JOB = 'MGR' AND
DEPT IN (SELECT DEPTNUMB FROM ORG
                    WHERE DIVISION = 'MIDWEST')
```

2. List the managers whose departments average salary is greater than the overall average salary. See ex82.sql

RESULTS:

DEPT	NAME	SALARY	JOB
10	MOLINARE	22059.2	MGR
10	LU	20010	MGR
10	DANIELS	19260.3	MGR
10	JONES	21234	MGR

```
SELECT DEPT, NAME, SALARY, JOB
FROM STAFF
WHERE JOB = 'MGR' AND
SELECT DEPT, AVG(SALARY) FROM STAFF GROUP BY DEPT
HAVING AVG(SALARY) >
(SELECT AVG(SALARY) FROM STAFF)
```

3. Find the employee with the highest salary. See ex83.sql

RESULTS:

NAME	SALARY
MOLINARE	22959.2

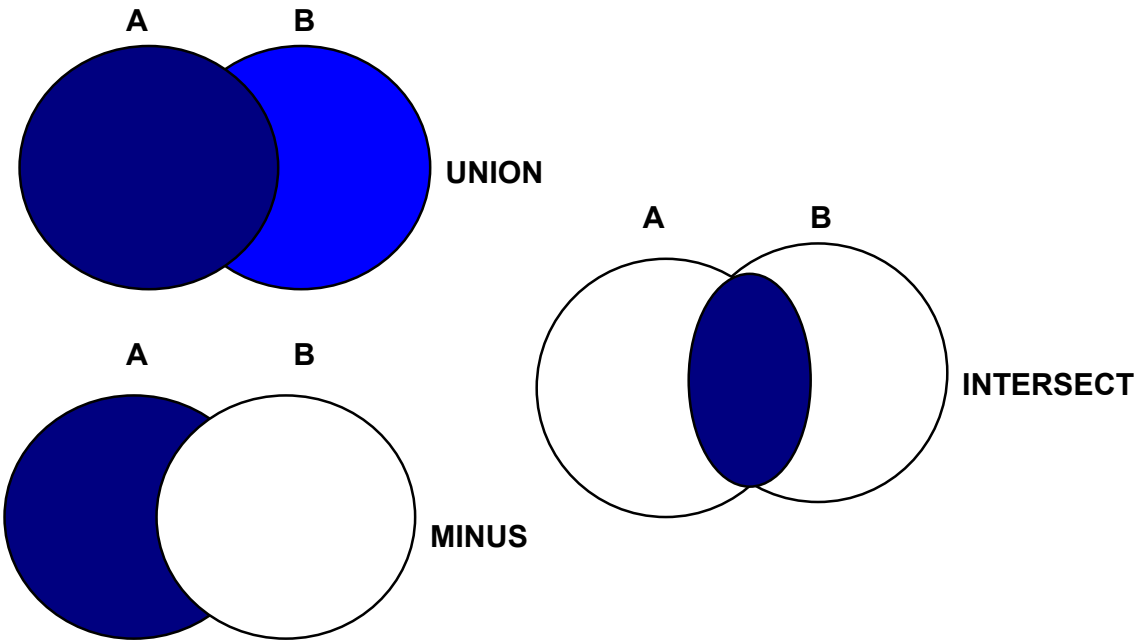
```
SELECT NAME, SALARY
FROM STAFF
WHERE SALARY =
(SELECT MAX(SALARY) FROM STAFF)
```

4. Retrieve the employee number, name, salary, and commission of all employees who earn more than $20,000.00 only if any employee earns less than $350.00 commission. (Hint: use an exist statement). See ex84.sql

RESULTS: 7 rows

```
SELECT ID, NAME, SALARY, COMM
FROM STAFF
WHERE SALARY + NVL(COMM,0) > 20000.00
AND EXISTS
(SELECT 1 FROM STAFF WHERE COMM < 350)
```

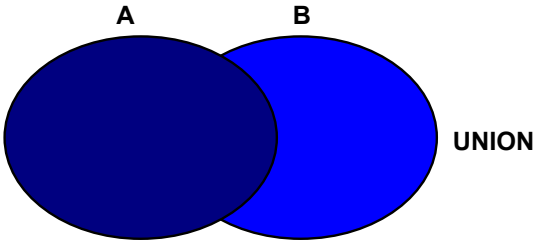
SET OPERATORS (Conjunctions)



Set operators combine the results of multiple queries into a single result. The following table lists SQL set operators in order of precedence from high to low.

Operator	Function	Example
UNION	Combines queries to return all distinct rows returned by any individual query (OR)	...SELECT .. UNION SELECT ...
INTERSECT	Combines queries to return all distinct rows returned by every individual query. (AND)	...SELECT.... INTERSECT SELECT..
MINUS	Combines queries to return all distinct rows returned by the first query but not the second query (AND NOT)	....SELECT.... MINUS SELECT.....

UNION



A **UNION** conjunction would combine rows in all queries in the union eliminating duplicate rows.

- The results would be similar to those returned when using a WHERE clause with an OR logical operator.

TASK: List employees by department. Designate for each employee if his years of service are from 0 to 5, from 6 to 10, or greater than 10

SQL:

```
SELECT DEPT, ID, NAME, '0-5 YEARS'
FROM STAFF
WHERE YEARS < 6
UNION
SELECT DEPT, ID, NAME ,'6-10 YEARS'
FROM STAFF
WHERE YEARS BETWEEN 6 AND 10
UNION
SELECT DEPT, ID, NAME, 'MORE THAN 10 YEARS'
FROM STAFF
WHERE YEARS >10
ORDER BY 1,2
```

RESULT:

DEPT	ID	NAME	0 - 5 YEARS
10	160	MOLINARE	6 - 10 YEARS
10	210	LU	6 - 10 YEARS
10	240	DANIELS	0 - 5 YEARS
10	260	JONES	MORE THAN 10 YRS
..	..	..	..

Exercise 9 on Union:

1. List all the applicants and staff members, order by ID. Label each person as either "STAFF" or "APPLICANT". See EX91.sql

   SELECT ID, NAME, JOB, 'STAFF'

   FROM STAFF

   UNION ALL

   SELECT TEMPID, NAME, COMMENTS,'APPLICANT'

   FROM APPLICANT

   RESULTS:

   44 or 45 rows selected

2. Provide an employee list which indicates which employees make less than OR EQUAL the average SALARY and which employees have salaries greater than or equal to the average SALARY. See EX92.sql

Select EMPNO, DEPT, NAME, SALARY, 'LESS THAN OR EQUAL TO AVG'

FROM STAFF

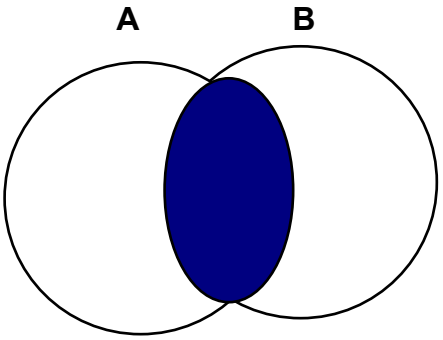
WHERE SALARY <= (SELECT AVG(SALARY) FROM STAFF)

UNION

RESULTS:

35 rows selected

SET OPERATOR INTERSECT



INTERSECT

The INTERSECT clause results would be similar to those returned when using a WHERE clause with an AND logical operator.

The following example would produce an empty results table because an employee is only assigned one DEPTNO at any given time

SELECT EMPNO, DEPTNO FROM EMP WHERE DEPTNO NOT IN

(SELECT EMPNO, DEPTNO FROM EMP WHERE DEPTNO = 003)

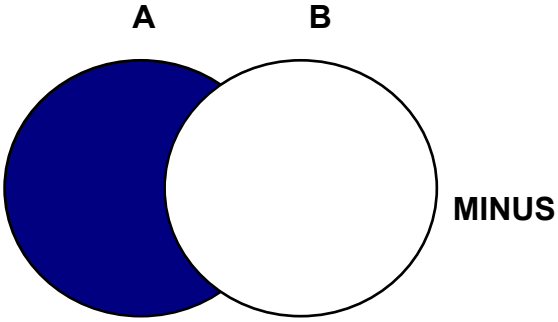
The following use of intersect would give a non-empty results table:

SELECT EMPNO FROM EMP
INTERSECT
SELECT EMPNO FROM PROJECTB

Results:

EMPNO
10
20

THE MINUS CONJUNCTION



The minus conjunction is similar to using the 'AND NOT' logical operator is a WHERE clause of a SELECT statement

SELECT EMPNO, SAL FROM EMP
WHERE SAL > 46000
MINUS
SELECT EMPNO, SAL FROM EMP
WHERE DEPTNO = 001

Results:

EMPNO	SAL
50	90000

SELECT EMPNO CONTRACTOR FROM PROJECTB
MINUS
SELECT EMPNO FROM EMP

Results:

CONTRACTOR
578
789

MODIFY Functions

SQL Insert a Row:

INSERT INTO ORG
VALUES (95, 'COASTAL', 150, 'MIDWEST','HOUSTON')

RESULT:
Adds the following row to the ORG Table

DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
95	COASTAL	150	MIDWEST	HOUSTON

Data value list must correspond to the list of columns in the table

Null values may be specified for a column with null

MODIFY FUNCTIONS

SQL Insert Selected Columns

```
INSERT INTO ORG(DEPTNUMB, DEPTNAME, LOCATION)
VALUES (95,'COASTAL', 'HOUSTON')
```

- Columns not specified are given the value null
- A value must be specified for any column defined as not null in the table create statement

```
INSERT INTO MYSTAFF
SELECT ID, NAME,DEPT
FROM STAFF WHERE JOB = 'CLERK'
```

Copies all clerk rows from STAFF to table MYSTAFF

The insert (as with all updates) may be performed on a table if

- You created the table
- You have been granted insert authority on the table

UPDATE A SINGLE ROW

Task: Update employee number 150 to change job description to Manager, Department to 95 and provide a 15% increase in salary.

SQL:

```
UPDATE STAFF
SET JOB='MGR', DEPT=95,  SALARY=SALARY * 1.15
WHERE ID=150
```

This example updates the department, job and salary for ID 150

Update multiple rows

SQL:

```
UPDATE STAFF
    SET COMM = NVL(COMM,0) + 500
    WHERE DEPT = 38
```

This query gives every employee in Department 38 a $500 commission bonus

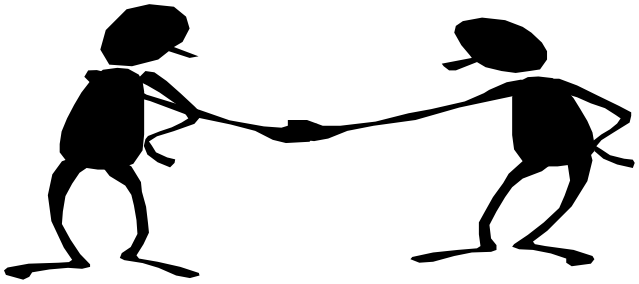
COMPLEX UPDATE

Task: Provide all employee who worked for Project number 50 a bonus equal to 15% of their salary. This requires that you search the Project table to find all employees in Project 50 and then update the employees Bonus field in the Staff table

```
Update Staff
Set Bonus =  salary * .15
where id in
(Select empno
 from project
where projectno = 50);
```

Result: All employees attached to Project number 50 are given a 15% bonus.

MERGING DATA IN ORACLE TABLES



With one statement we can either insert or update

The merge command allows you to decide based upon a condition whether you will be inserting or updating. Implicit in the use of the merge command is the concept that you are merging the contents of one table into another based on whether the values exist in the second table, extending the table join principles reviewed earlier. For example, assume that a transaction file being read in PLSQL has both new employees salaries and old employees salaries that require updating. New employees are not immediately assigned department numbers. So we can use the MERGE command to decipher whether we are going to insert new employees or simply update them.

```
MERGE into staff s1
Using org o2 on (s1.dept = o2.deptnumb )
        When matched then update set s1.salary = :new_salary
        When not matched then insert (id,name,dept,years,job,salary,comm.)
        Values (:new_id,:new_name,:new_dept,:new_years,
                :new_salary,:new_comm.);
```

The above example states in line 1 “Merge rows in the STAFF table”. The second line specifies “to match the data using the Org and Staff tables). The third and fourth lines define that if a employee has a department number in both tables then his/her salary is updated otherwise the NOT MATCHED says they don’t exist and can be inserted**.**

SQL DELETE

Single row delete

1. *DELETE FROM STAFF*
   WHERE ID = 150

Multiple row delete

2. *DELETE FROM STAFF*
   WHERE JOB = 'CLERK'

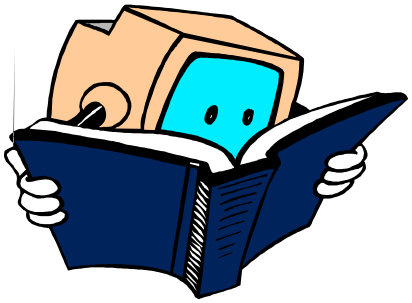
COMPLEX DELETE

Task: Delete all employee who worked for Project number 50.

```
Delete from  Staff
where id in
(Select empno
 from project
where projectno = 50);
```

Result: All employees attached to Project number 50 are deleted

LAB 4 - SELECTING ROWS FROM MORE THAN ONE TABLE USE THE ORG, STAFF AND APPLICANT UNLESS OTHERWISE IDENTIFIED.



1. TASK: Retrieve all employees in department 20. Print out the department name along with their id, name, and salary.

2. TASK: Select the employee number, name, job, salary, department and division for all employees whose salary is in the range of 15000 to 30000 dollars. Sort the data by division and job. See salrang.sql

   RESULTS:

   23 rows are selected

3. TASK: Display the rows in staff table that do not have a match in the org table (an employee who does not have a department number in the org table) as well as those employees with department numbers that do have a match in the org table. See outrjoin.sql. Optional - See if you can just return a staff row(s) that does not have a match in the org table.

   RESULTS:

   You will have 35 rows returned which include an employee named JAMES who is a clerk with no valid department number.

LAB 4 - SELECTING ROWS FROM MORE THAN ONE TABLE

4. a. Insert your name and relevant information into the applicant table.

   b. Insert a row into the org table which creates a new Division (Southwest), department number (77), department name (Dallas), location (Plano), and manager (10).

   c. You have been hired by the company. Move your applicant information from the applicant table to the staff table. Your salary is $50,000.00, you have no commission, and your job status is “MGR”. see cmplxins.sql

5. Display all location or division or department names eliminating duplicates. The company is looking at defining a new set of specifications on the names it uses so it needs to look at all the current alphanumeric names being used. Eliminate duplicates. See union.sql

6. You have received a promotion. Upgrade your salary by 10%. Give yourself a $5,000.00 commission as well. See update6.sql

7. Find the person(s) who has worked with the company longest and give them a 10% salary increase. See update7.sql

8. Whoops!!. Our applicant tables need to be cleaned up. Delete anyone in the applicant tables who is now a permanent employee. Please delete them from the appropriate tables. See cmplxdel.sql