

PRACTICE ON SQL

ALTER TABLE AP for

adding constraint, modifying column sizes, setting NULL/NOT NULL, DEFAULT values

- Adding CHECK constraints to enforce business rules

```
alter table AP_EMP add constraint C_EMP_EMPNO check (EMPNO between 1000 and 9999);
```

```
alter table AP_PROJECT add constraint C_PROJECT_PROJID check (PROJID between 100 and 999);
```

```
alter table AP_DEPT add constraint C_DEPT_DEPTNO check (DEPTNO between 10 and 99);
```

```
alter table AP_EMP add constraint C_EMP_ELNAME check (ELNAME=upper(ELNAME));
```

- Test constraint enforcements

```
insert into AP_EMP values (10001, 'Robert', 'Smith', 'TEMP', 7369, sysdate-40, 1000, null, 10);
```

```
insert into AP_EMP values (9999, 'Robert', 'Smith', 'TEMP', 7369, sysdate-40, 1000, null, 10);
```

- Modify column to add DEFAULT value

```
alter table AP_PROEMP modify ( HOURS default 0 );
```

- Modify column to allow for NULL

```
alter table AP_EMP modify job NULL;
```

- Modify column to change size and set NOT NULL

```
alter table AP_EMP modify job varchar2(20) NOT NULL;
```

- Adding foreign key constrain

```
alter table AP_EMP add constraint FK_EMP_MGR foreign key (MGR) references AP_EMP(EMPNO);
```

ALTER TABLE AP_to add/remove column

```
alter table AP_EMP add BIRTHDATE date;
```

```
alter table AP_EMP add ADDRESS varchar2(50);
```

```
alter table AP_EMP drop ( BIRTHDATE, ADDRESS);
```

Creating table from other table

CREATE TABLE AP_EMPTEST AS SELECT * FROM AP_EMP -- CREATE A TABLE FROM EXISTING TABLE

SELECT * FROM AP_EMPTEST -- RETRIVING/SELECTING ALL RECORDS OF A TABLE

TRUNCATE TABLE AP_EMPTEST -- REMOVING/DELETING ALL RECORDS OF A TABLE

SELECT * FROM AP_EMPTEST-- RETRIVING/SELECTING ALL RECORDS OF A TABLE

DROP TABLE AP_EMPTEST -- DROPPING A TABLE

SELECT * FROM AP_EMPTEST -- RETRIVING/SELECTING ALL RECORDS OF A TABLE

Data Manipulation Language

- A DML statement is executed when you:
 - Add new rows to a table
 - Modify existing rows in a table
 - Remove existing rows from a table
- A transaction consists of a collection of DML statements that form a logical unit of work.

– CREATING A EMPTY TABLE FROM ANOTHER TABLE;

```
CREATE TABLE AP_DEPTTEST AS SELECT * FROM AP_DEPT WHERE 1=2;
```

```
CREATE TABLE AP_EMPTEST AS SELECT * FROM AP_EMP WHERE 1=2;
```

--POPULATING TABLE FROM ANOTHER TABLE;

```
INSERT INTO AP_DEPTTEST SELECT * FROM AP_DEPT;
```

```
INSERT INTO AP_EMPTEST SELECT * FROM AP_EMP;
```

```
COMMIT;
```

```
INSERT INTO AP_DEPTTEST (deptno,dname,loc) VALUES ('TRAINING','AUSTIN');
```

```
INSERT INTO AP_DEPTTEST (deptno,dname,loc) VALUES (NULL,'TRAINING','AUSTIN');
INSERT INTO AP_DEPTTEST (deptno,dname,loc) VALUES (400,'TRAINING','AUSTIN');
INSERT INTO AP_DEPTTEST VALUES (50,'TRAINING','AUSTIN');
SELECT * FROM AP_DEPTTEST;
```

```
COMMIT;
```

```
INSERT INTO AP_DEPTTEST VALUES (60,'LEGAL',NULL);
SELECT * FROM AP_DEPTTEST;
```

The UPDATE Statement Syntax

- **Modify existing rows with the UPDATE statement.**

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

- **Update more than one row at a time, if required.**

```
SELECT deptno
FROM ap_emp
WHERE empno=7369;
UPDATE ap_emptest
SET deptno=30
WHERE empno=7369;
```

```
SELECT empno,deptno  
FROM ap_emptest;
```

```
UPDATE ap_emptest  
SET sal=sal+100  
WHERE job='CLERK';
```

```
UPDATE ap_emptest  
SET comm=comm +100;
```

```
UPDATE ap_emptest  
SET comm=nvl(comm,0)+100 , ELNAME=INITCAP(ELNAME);
```

```
SELECT * FROM ap_emptest;
```

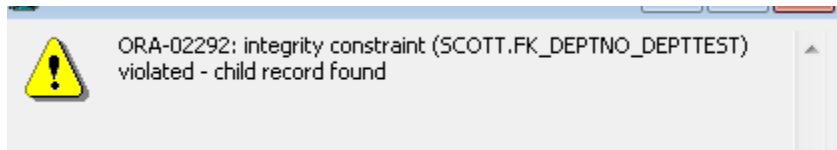
```
commit;
```

The DELETE Statement

You can remove existing rows from a table by using the DELETE statement.

```
DELETE [FROM]    table  
[WHERE           condition];
```

DELETE FROM depttest;



DELETE FROM ap_emptest
WHERE deptno=30;

DELETE FROM ap_emptest;

SELECT COUNT(*) FROM ap_emptest;

SQL Statements

Oracle SQL complies with industry-accepted standards. Oracle Corporation ensures future compliance with evolving standards by actively involving key personnel in SQL standards committees. Industry-accepted committees are the American National Standards Institute (ANSI) and the International Standards Organization (ISO). Both ANSI and ISO have accepted SQL as the standard language for relational databases.

Statement	Description
SELECT	Retrieves data from the database
INSERT UPDATE DELETE MERGE	Enters new rows, changes existing rows, and removes unwanted rows from tables in the database, respectively. Collectively known as <i>data manipulation language (DML)</i> .
CREATE ALTER DROP RENAME TRUNCATE	Set up, change, and remove data structures from tables. Collectively known as <i>data definition language (DDL)</i> .
COMMIT ROLLBACK SAVEPOINT	Manage the changes made by DML statements. Changes to the data can be grouped together into logical transactions.
GRANT REVOKE	Give or remove access rights to both the Oracle database and the structures within it. Collectively known as <i>data control language (DCL)</i> .

Capabilities of SQL SELECT Statements

Projection

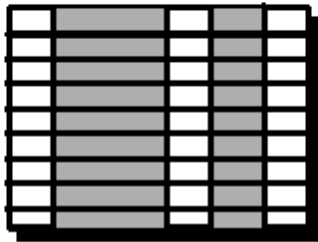


Table 1

Selection

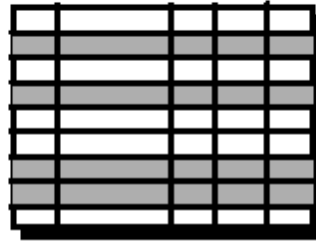


Table 1

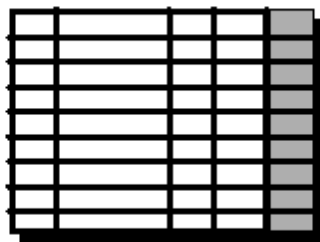


Table 1

Join

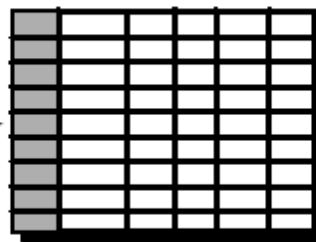


Table 2

ORACLE

Basic SELECT structure

Clauses of SELECT statement

SELECT < column list, expressions, literals>

FROM <table list>

WHERE <filter conditions with AND/OR/NOT logical operators>

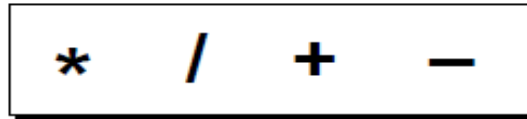
GROUP BY <column list for aggregate functions COUNT/SUM/MIN/MAX/AVG etc.>

HAVING <filter conditions for grouping results>

ORDER BY <column list for sorting result set>

(SELECT and FORM clause are mandatory, all other clauses are optional and to be used as required by query result)

Operator Precedence



- Multiplication and division take priority over addition and subtraction.
- Operators of the same priority are evaluated from left to right.
- Parentheses are used to force prioritized evaluation and to clarify statements.

```
SELECT *  
FROM ap_emp;
```

```
SELECT empno, ELNAME, job, sal  
FROM ap_emp;
```

```
SELECT empno "Employee Number", ELNAME as name , job, sal "Monthly Salary USD"  
FROM ap_emp;
```

```
SELECT ELNAME || ' was hired on ' || hiredate || ' and current monthly salary is USD' || sal  
FROM ap_emp;
```

```
SELECT A.ELNAME, A.sal, A.sal+100, A.comm, A.comm+10  
from ap_emp A;
```

```
SELECT A.ELNAME, A.sal, A.sal+100, A.comm, nvl(A.comm,0)+10  
FROM ap_emp A;
```

```
SELECT DISTINCT job  
FROM ap_emp;
```

```
SELECT DISTINCT job,deptno  
FROM ap_emp;
```

```
SELECT DISTINCT job,deptno  
FROM ap_emp  
ORDER BY job, deptno;
```

```
SELECT DISTINCT job,deptno  
FROM ap_emp  
ORDER BY deptno, job;
```

```
SELECT DISTINCT job,deptno  
FROM ap_emp  
ORDER BY deptno, job desc;
```

```
SELECT ELNAME, sal  
FROM ap_emp
```

ORDER BY 2;

SELECT ELNAME, sal
FROM ap_emp
ORDER BY 2 desc, 1;

WHERE clause (SELECTION of results - Limiting result sets based upon the conditions)

Comparison Conditions

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

SELECT *

```
FROM ap_emp  
WHERE sal>9000;
```

```
SELECT *  
FROM ap_emp  
WHERE sal<9000;
```

```
SELECT ELNAME, sal AS salary  
FROM ap_emp  
WHERE sal>=9000  
ORDER BY salary;
```

```
SELECT ELNAME, deptno  
FROM ap_emp  
WHERE deptno=10  
ORDER BY 1;
```

```
SELECT ELNAME, deptno  
FROM ap_emp  
WHERE deptno<>10  
ORDER BY 2, 1;
```

Other Comparison Conditions

Operator	Meaning
BETWEEN ...AND...	Between two values (inclusive)
IN (set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

```
SELECT ELNAME, sal
FROM ap_emp
WHERE sal BETWEEN 4500 AND 9000
ORDER BY sal;
```

```
SELECT ELNAME, deptno
FROM ap_emp
WHERE deptno IN (10,30)
ORDER BY 2;
```

```
SELECT ELNAME, deptno
FROM ap_emp
```

```
WHERE ELNAME like 'A%'
ORDER BY 1;
```

```
SELECT ELNAME, deptno
FROM ap_emp
WHERE ELNAME like '%N'
ORDER BY 1;
```

```
SELECT ELNAME, deptno
FROM ap_emp
WHERE ELNAME like '%A%'
ORDER BY 1;
```

```
SELECT ELNAME, sal, comm
FROM ap_emp
WHERE comm IS NULL;
```

```
SELECT ELNAME, sal, comm
FROM ap_emp
WHERE comm IS NOT NULL;
```


Logical Conditions

Operator	Meaning
AND	Returns TRUE if <i>both</i> component conditions are true
OR	Returns TRUE if <i>either</i> component condition is true
NOT	Returns TRUE if the following condition is false

```
SELECT ELNAME, sal, deptno  
FROM ap_emp  
WHERE sal>5000 AND deptno=10;
```

```
SELECT ELNAME, sal, deptno  
FROM ap_emp  
WHERE sal>5000 OR deptno=10;
```

```
SELECT ELNAME, sal, deptno, job  
FROM ap_emp  
WHERE sal<5000 OR deptno=10 AND job='ANALYST';
```

```
SELECT ELNAME, sal, deptno, job
```

```
FROM ap_emp
WHERE (sal<5000 OR deptno=10) AND job='ANALYST';
```

```
SELECT ELNAME,sal, deptno, job
FROM ap_emp
WHERE sal<5000 OR deptno=10 OR job='ANALYST';
```

```
SELECT ELNAME, sal
FROM ap_emp
WHERE sal NOT BETWEEN 4500 AND 9000
ORDER BY sal;
```

```
SELECT ELNAME,deptno
FROM ap_emp
WHERE deptno NOT IN (10,30)
ORDER BY 2;
```

```
SELECT ELNAME, deptno
FROM ap_emp
WHERE ELNAME NOT like 'A%'
ORDER BY 1;
```

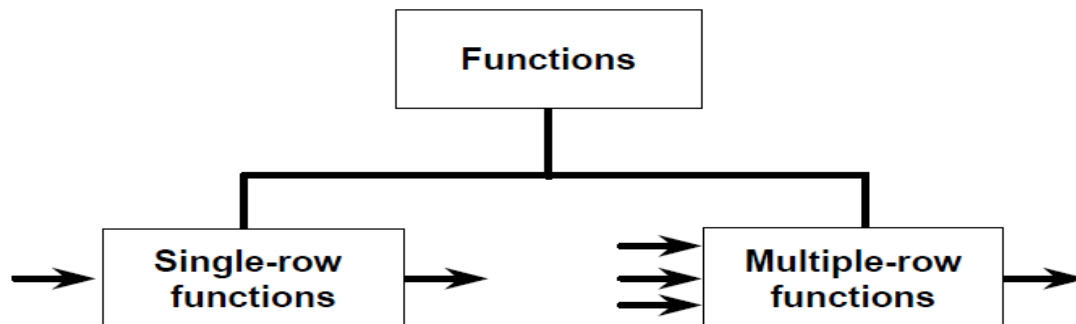
SQL Functions

Functions are a very powerful feature of SQL and can be used to do the following:

- Perform calculations on data
- Modify individual data items
- Manipulate output for groups of rows
- Format dates and numbers for display
- Convert column data types

SQL functions sometimes take arguments and always return a value.

Two Types of SQL Functions



ORACLE SQL FUCTIONS

<https://docs.oracle.com/database/121/SQLRF/functions.htm#SQLRF006>

```
SELECT ELNAME, lower(ELNAME), initcap(ELNAME), substr(ELNAME, 1,2), substr(ELNAME, -3,2), substr(ELNAME,  
-2), lower(substr( ELNAME, 1,2))  
FROM ap_emp  
WHERE UPPER(JOB)='CLERK';
```

```
SELECT ELNAME, LENGTH(ELNAME), sal, LPAD(sal, 10,0)  
FROM ap_emp  
WHERE UPPER(JOB)='CLERK';
```

Number Functions

- **ROUND:** Rounds value to specified decimal

`ROUND (45.926, 2)`  `45.93`

- **TRUNC:** Truncates value to specified decimal

`TRUNC (45.926, 2)`  `45.92`

- **MOD:** Returns remainder of division

`MOD (1600, 300)`  `100`

0

```
SELECT ROUND(45.926,2), ROUND( 45.926) FROM dual;
```

```
SELECT TRUNC(45.926,2), TRUNC( 45.926) FROM dual;
```

```
SELECT ELNAME, sal, MOD(sal, 1000)  
FROM ap_emp;
```

Arithmetic with Dates

- Add or subtract a number to or from a date for a resultant date value.
- Subtract two dates to find the number of days between those dates.
- Add hours to a date by dividing the number of hours by 24.

```
SELECT sysdate  
FROM DUAL;
```

```
SELECT sysdate+14  
FROM DUAL;
```

```
SELECT SYSDATE-30  
FROM DUAL;
```

```
SELECT ELNAME, hiredate, sysdate-hiredate "No of days at work"  
FROM ap_emp;
```

```
SELECT ELNAME, hiredate, (sysdate-hiredate)/365 "No of years at work"  
FROM ap_emp;
```

```
SELECT ELNAME, hiredate, trunc( (sysdate-hiredate)/365 ) "No of years at work"  
FROM ap_emp;
```

```
SELECT ELNAME, hiredate, round( (sysdate-hiredate)/7) "No of weeks at work"  
FROM ap_emp;
```

Date Functions

Function	Description
MONTHS_BETWEEN	Number of months between two dates
ADD_MONTHS	Add calendar months to date
NEXT_DAY	Next day of the date specified
LAST_DAY	Last day of the month
ROUND	Round date
TRUNC	Truncate date

Using Date Functions

- MONTHS_BETWEEN ('01-SEP-95', '11-JAN-94')
→ 19.6774194
- ADD_MONTHS ('11-JAN-94', 6) → '11-JUL-94'
- NEXT_DAY ('01-SEP-95', 'FRIDAY')
→ '08-SEP-95'
- LAST_DAY('01-FEB-95') → '28-FEB-95'

Using Date Functions

Assume SYSDATE = '25-JUL-95':

- ROUND(SYSDATE, 'MONTH') → 01-AUG-95
- ROUND(SYSDATE, 'YEAR') → 01-JAN-96
- TRUNC(SYSDATE, 'MONTH') → 01-JUL-95
- TRUNC(SYSDATE, 'YEAR') → 01-JAN-95

Using the TO_CHAR Function with Dates

```
TO_CHAR(date, 'format_model')
```

The format model:

- Must be enclosed in single quotation marks and is case sensitive
- Can include any valid date format element
- Has an *fm* element to remove padded blanks or suppress leading zeros
- Is separated from the date value by a comma

Elements of the Date Format Model

YYYY	Full year in numbers
YEAR	Year spelled out
MM	Two-digit value for month
MONTH	Full name of the month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month

Elements of the Date Format Model

- Time elements format the time portion of the date.

HH24:MI:SS AM	15:45:32 PM
---------------	-------------

- Add character strings by enclosing them in double quotation marks.

DD "of" MONTH	12 of OCTOBER
---------------	---------------

- Number suffixes spell out numbers.

ddspth	fourteenth
--------	------------

```
SELECT to_char(sysdate, 'dd-mon-yy hh24:mi:ss ')  
FROM dual;
```

```
SELECT to_char(sysdate, 'dd-mon-yy hh:mi:ss AM')  
FROM dual;
```

```
SELECT to_char(sysdate, 'dd-MON-yyyy hh:mi:ss AM')  
FROM dual;
```

```
SELECT to_char(sysdate, 'dd-Month,yyyy hh:mi:ss AM')  
FROM dual;
```

```
SELECT to_char(sysdate, 'ddspth -Month,yyyy hh:mi:ss AM')
```

```
FROM dual;
```

```
SELECT to_char(sysdate, 'DdsptH -Month,yyyy hh:mi:ss AM')  
FROM dual;
```

```
SELECT to_char(sysdate, 'DdsptH -Month "of" Year hh:mi:ss AM')  
FROM dual;
```

```
SELECT to_char (sysdate, 'DDD')  
FROM dual;
```

```
SELECT to_char (sysdate, 'DD')  
FROM dual;
```

```
SELECT to_char (sysdate, 'D')  
FROM dual;
```

```
SELECT to_char (sysdate, 'DY')  
FROM dual;
```

```
SELECT to_char (sysdate, 'Day')  
FROM dual;
```

```
SELECT to_char (sysdate, 'WW')  
FROM dual;
```

```
SELECT to_char (sysdate, 'W')
```

FROM dual;

Using the TO_CHAR Function with Numbers

```
TO_CHAR(number, 'format_model')
```

These are some of the format elements you can use with the TO_CHAR function to display a number value as a character:

9	Represents a number
0	Forces a zero to be displayed
\$	Places a floating dollar sign
L	Uses the floating local currency symbol
.	Prints a decimal point
,	Prints a thousand indicator

```
SELECT ELNAME, sal, to_char(sal, '99999'),  
       to_char(sal, '09999.99'),  
       to_char(sal, '$99999'),  
       to_char(sal, '$99,999'),  
       to_char(sal, 'L99999')  
FROM ap_emp;
```

What Are Group Functions?

Group functions operate on sets of rows to give one result per group.

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2500
50	2500
...	
110	12000
110	8300

20 rows selected.

The maximum salary in the EMPLOYEES table.

MAX(SALARY)
24000

ORACLE

Types of Group Functions

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

```
SELECT COUNT(*), MIN(sal), MAX(sal), AVG(sal), SUM(sal)
FROM ap_emp
WHERE deptno=10;
```

```
SELECT MIN(hiredate), MAX(hiredate)
FROM ap_emp
WHERE deptno=20;
```

```
SELECT COUNT( DISTINCT job)
FROM ap_emp
WHERE deptno=20;
```

```
SELECT AVG(comm)
FROM ap_emp;
```

```
SELECT AVG(NVL(comm,0))
from ap_emp;
```

```
SELECT deptno, SUM(sal)
FROM ap_emp
GROUP BY deptno
ORDER BY 2;
```

```
SELECT deptno, job, AVG(sal)
FROM ap_emp
GROUP BY deptno, job
ORDER BY 1,2;
```

```
SELECT deptno, job, AVG(sal)
FROM ap_emp
GROUP BY deptno, job
HAVING AVG(sal)>3000
ORDER BY 1,2;
```

Joining Tables Using Oracle Syntax

Use a join to query data from more than one table.

```
SELECT  table1.column, table2.column  
FROM    table1, table2  
WHERE   table1.column1 = table2.column2;
```

- Write the join condition in the WHERE clause.
- Prefix the column name with the table name when the same column name appears in more than one table.

Guidelines

- When writing a SELECT statement that joins tables, precede the column name with the table name for clarity and to enhance database access.
- If the same column name appears in more than one table, the column name must be prefixed with the table name.
- To join n tables together, you need a minimum of $n-1$ join conditions. For example, to join four tables, a minimum of three joins is required. This rule may not apply if your table has a concatenated primary key, in which case more than one column is required to uniquely identify each row.

What Is an Equijoin?

EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
200	10
201	20
202	20
124	50
141	50
142	50
143	50
144	50
103	60
104	60
105	60
106	60
107	60
108	60
109	60
110	60
111	60
112	60
113	60
114	60
115	60
116	60
117	60
118	60
119	60
120	60
121	60
122	60
123	60
124	60
125	60
126	60
127	60
128	60
129	60
130	60
131	60
132	60
133	60
134	60
135	60
136	60
137	60
138	60
139	60
140	60
141	60
142	60
143	60
144	60
145	60
146	60
147	60
148	60
149	60
150	60
151	60
152	60
153	60
154	60
155	60
156	60
157	60
158	60
159	60
160	60
161	60
162	60
163	60
164	60
165	60
166	60
167	60
168	60
169	60
170	60
171	60
172	60
173	60
174	60
175	60
176	60
177	60
178	60
179	60
180	60
181	60
182	60
183	60
184	60
185	60
186	60
187	60
188	60
189	60
190	60
191	60
192	60
193	60
194	60
195	60
196	60
197	60
198	60
199	60
200	60
201	60
202	60
203	60
204	60
205	60
206	60
207	60
208	60
209	60
210	60
211	60
212	60
213	60
214	60
215	60
216	60
217	60
218	60
219	60
220	60
221	60
222	60
223	60
224	60
225	60
226	60
227	60
228	60
229	60
230	60
231	60
232	60
233	60
234	60
235	60
236	60
237	60
238	60
239	60
240	60
241	60
242	60
243	60
244	60
245	60
246	60
247	60
248	60
249	60
250	60
251	60
252	60
253	60
254	60
255	60
256	60
257	60
258	60
259	60
260	60
261	60
262	60
263	60
264	60
265	60
266	60
267	60
268	60
269	60
270	60
271	60
272	60
273	60
274	60
275	60
276	60
277	60
278	60
279	60
280	60
281	60
282	60
283	60
284	60
285	60
286	60
287	60
288	60
289	60
290	60
291	60
292	60
293	60
294	60
295	60
296	60
297	60
298	60
299	60
300	60
301	60
302	60
303	60
304	60
305	60
306	60
307	60
308	60
309	60
310	60
311	60
312	60
313	60
314	60
315	60
316	60
317	60
318	60
319	60
320	60
321	60
322	60
323	60
324	60
325	60
326	60
327	60
328	60
329	60
330	60
331	60
332	60
333	60
334	60
335	60
336	60
337	60
338	60
339	60
340	60
341	60
342	60
343	60
344	60
345	60
346	60
347	60
348	60
349	60
350	60
351	60
352	60
353	60
354	60
355	60
356	60
357	60
358	60
359	60
360	60
361	60
362	60
363	60
364	60
365	60
366	60
367	60
368	60
369	60
370	60
371	60
372	60
373	60
374	60
375	60
376	60
377	60
378	60
379	60
380	60
381	60
382	60
383	60
384	60
385	60
386	60
387	60
388	60
389	60
390	60
391	60
392	60
393	60
394	60
395	60
396	60
397	60
398	60
399	60
400	60
401	60
402	60
403	60
404	60
405	60
406	60
407	60
408	60
409	60
410	60
411	60
412	60
413	60
414	60
415	60
416	60
417	60
418	60
419	60
420	60
421	60
422	60
423	60
424	60
425	60
426	60
427	60
428	60
429	60
430	60
431	60
432	60
433	60
434	60
435	60
436	60
437	60
438	60
439	60
440	60
441	60
442	60
443	60
444	60
445	60
446	60
447	60
448	60
449	60
450	60
451	60
452	60
453	60
454	60
455	60
456	60
457	60
458	60
459	60
460	60
461	60
462	60
463	60
464	60
465	60
466	60
467	60
468	60
469	60
470	60
471	60
472	60
473	60
474	60
475	60
476	60
477	60
478	60
479	60
480	60
481	60
482	60
483	60
484	60
485	60
486	60
487	60
488	60
489	60
490	60
491	60
492	60
493	60
494	60
495	60
496	60
497	60
498	60
499	60
500	60
501	60
502	60
503	60
504	60
505	60
506	60
507	60
508	60
509	60
510	60
511	60
512	60
513	60
514	60
515	60
516	60
517	60
518	60
519	60
520	60
521	60
522	60
523	60
524	60
525	60
526	60
527	60
528	60
529	60
530	60
531	60
532	60
533	60
534	60
535	60
536	60
537	60
538	60
539	60
540	60
541	60
542	60
543	60
544	60
545	60
546	60
547	60
548	60
549	60
550	60
551	60
552	60
553	60
554	60
555	60
556	60
557	60
558	60
559	60
560	60
561	60
562	60
563	60
564	60
565	60
566	60
567	60
568	60
569	60
570	60
571	60
572	60
573	60
574	60
575	60
576	60
577	60
578	60
579	60
580	60
581	60
582	60
583	60
584	60
585	60
586	60
587	60
588	60
589	60
590	60
591	60
592	60
593	60
594	60
595	60
596	60
597	60
598	60
599	60
600	60
601	60
602	60
603	60
604	60
605	60
606	60
607	60
608	60
609	60
610	60
611	60
612	60
613	60
614	60
615	60
616	60
617	60
618	60
619	60
620	60
621	60
622	60
623	60
624	60
625	60
626	60
627	60
628	60
629	60
630	60
631	60
632	60
633	60
634	60
635	60
636	60
637	60
638	60
639	60
640	60
641	60
642	60
643	60
644	60
645	60
646	60
647	60
648	60
649	60
650	60
651	60
652	60
653	60
654	60
655	60
656	60
657	60
658	60
659	60
660	60
661	60
662	60
663	60
664	60
665	60
666	60
667	60
668	60
669	60
670	60
671	60
672	60
673	60
674	60
675	60
676	60
677	60
678	60
679	60
680	60
681	60
682	60
683	60
684	60
685	60
686	60
687	60
688	60
689	60
690	60
691	60
692	60
693	60
694	60
695	60
696	60
697	60
698	60
699	60
700	60
701	60
702	60
703	60
704	60
705	60
706	60

```

SELECT a.empno, a.ELNAME,a.deptno, b.dname
FROM ap_emp a INNER JOIN ap_dept b ON a.deptno=b.deptno
order by 3
;

```

```

SELECT a.empno, a.ELNAME,a.deptno, b.dname
FROM ap_emp a JOIN ap_dept b ON a.deptno=b.deptno
order by 3
;

```

```

SELECT a.empno, a.projid,b.pname,a.hours
FROM ap_proemp a JOIN ap_project b ON a.projid=b.projid
WHERE a.hours>50
ORDER BY 1
;

```

```

SELECT a.empno, a.ELNAME,a.deptno, b.dname,b.loc,c.hours, d.pname
FROM ap_emp a JOIN ap_dept b ON a.deptno=b.deptno JOIN ap_proemp c ON a.empno=c.empno JOIN
ap_project d ON c.projid=d.projid
WHERE b.loc IN ('NEW YORK', 'DALLAS') AND c.hours>40
ORDER BY 1;

```

NATURAL JOIN: when FK and PK column of same name, do not need to write join condition.

SELECT empno, ELNAME,deptno, dname – column alias on common column will return error.
 FROM ap_emp NATURAL JOIN ap_dept;

NON-EQUIJOIN: Join based on other than equality operator

```
SELECT a.empno, a.ELNAME, a.sal, b.grade, b.losal, b.hisal
FROM ap_emp a, ap_salgrade b
WHERE a.sal between b.losal AND b.hisal
ORDER BY b.grade;
```

Outer Joins Syntax

- You use an outer join to also see rows that do not meet the join condition.
- The outer join operator is the plus sign (+).

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column = table2.column(+);
```

```
SELECT a.empno, a.ELNAME, b.deptno, b.dname
FROM ap_emp a RIGHT OUTER JOIN ap_dept b ON a.deptno=b.deptno
ORDER BY a.deptno
;
```

```
SELECT b.empno, b.ELNAME, a.deptno, a.dname
FROM ap_dept a LEFT OUTER JOIN ap_emp b ON a.deptno=b.deptno
ORDER BY b.deptno
```

;

```
SELECT b.empno, b.ELNAME,a.deptno,a.dname
FROM ap_dept a FULL OUTER JOIN ap_emp b ON a.deptno=b.deptno
ORDER BY b.deptno;
```

SELF JOIN

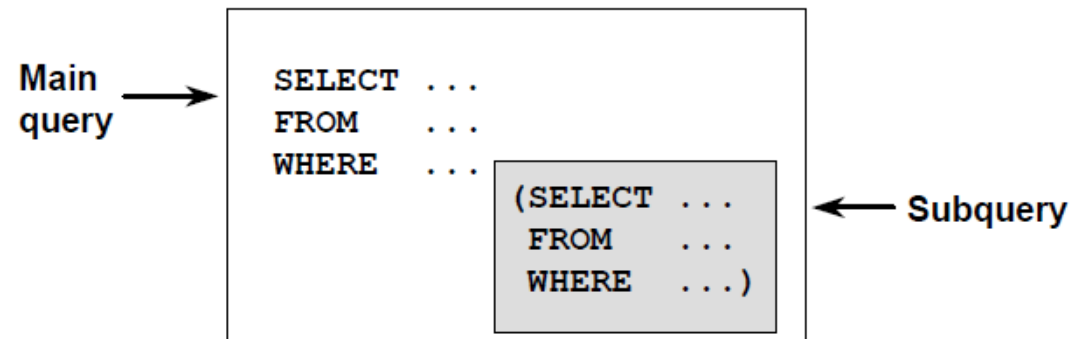
AP_EMP --- a					AP_EMP --- b			
EMPNO	ENAME	JOB	MGR		EMPNO	ENAME	JOB	MGR
7369	SMITH	CLERK	7902		7369	SMITH	CLERK	7902
7499	ALLEN	SALESMAN	7698		7499	ALLEN	SALESMAN	7698
7521	WARD	SALESMAN	7698		7521	WARD	SALESMAN	7698
7566	JONES	MANAGER	7839		7566	JONES	MANAGER	7839
7654	MARTIN	SALESMAN	7698		7654	MARTIN	SALESMAN	7698
7698	BLAKE	MANAGER	7839		7698	BLAKE	MANAGER	7839
7782	CLARK	MANAGER	7839		7782	CLARK	MANAGER	7839
7788	SCOTT	ANALYST	7566		7788	SCOTT	ANALYST	7566
7839	KING	PRESIDENT			7839	KING	PRESIDENT	
7844	TURNER	SALESMAN	7698		7844	TURNER	SALESMAN	7698
7876	ADAMS	CLERK	7788		7876	ADAMS	CLERK	7788
7900	JAMES	CLERK	7698		7900	JAMES	CLERK	7698
7902	FORD	ANALYST	7566		7902	FORD	ANALYST	7566
7934	MILLER	CLERK	7782		7934	MILLER	CLERK	7782

```
SELECT a.empno,a.ELNAME,a.mgr, b.empno "Manager ID", b.ELNAME "Manager Name"
FROM ap_emp a JOIN ap_emp b ON a.mgr=b.empno;
```

USING SUBQUERY

What Is a Subquery?

A subquery is a **SELECT** statement embedded in a clause of another **SQL** statement.



Single-Row Subqueries

- Return only one row
- Use single-row comparison operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

ORACLE

```
SELECT ELNAME, sal  
FROM ap_emp  
WHERE sal > (SELECT sal FROM ap_emp WHERE ELNAME='ALLEN');
```

```
SELECT ELNAME, job  
FROM ap_emp  
WHERE deptno=(SELECT deptno FROM ap_dept WHERE loc='NEW YORK');
```

```
SELECT ELNAME, job, sal
FROM ap_emp
WHERE sal > (SELECT avg(sal) FROM ap_emp);
```

```
SELECT ELNAME, job, sal
FROM ap_emp
WHERE sal > ( SELECT avg(sal) FROM ap_emp WHERE
              deptno= (SELECT deptno FROM ap_dept where loc ='NEW YORK')
              )
```

- Find out employees and their salary for those employees whose salary is higher than the average salary of employees working at NEW YORK.

```
SELECT job, AVG(sal)
FROM ap_emp
GROUP BY job
HAVING AVG(sal) = (SELECT MIN(AVG(sal)) FROM ap_emp GROUP BY job);
```

```
SELECT ELNAME, sal
FROM ap_emp
WHERE sal > ( SELECT avg(sal) FROM ap_emp GROUP BY deptno);
```



ORA-01427: single-row subquery returns more than one row

Multiple-Row Subqueries

- Return more than one row
- Use multiple-row comparison operators

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

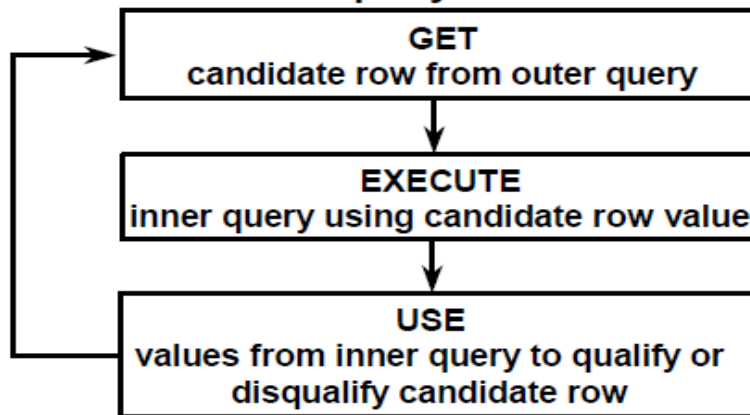
```
SELECT ELNAME, sal
FROM ap_emp
WHERE sal > ALL ( SELECT avg(sal) FROM ap_emp GROUP BY deptno);
```

```
SELECT ELNAME, sal
FROM ap_emp
WHERE sal > ANY ( SELECT avg(sal) FROM ap_emp GROUP BY deptno);
```

```
SELECT ELNAME, sal
FROM ap_emp
WHERE sal IN ( SELECT avg(sal) FROM ap_emp GROUP BY deptno);
```


Correlated Subqueries

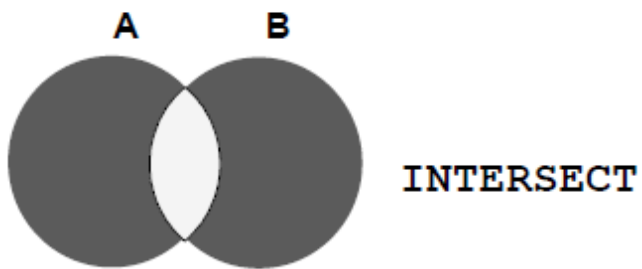
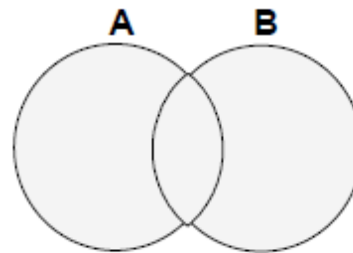
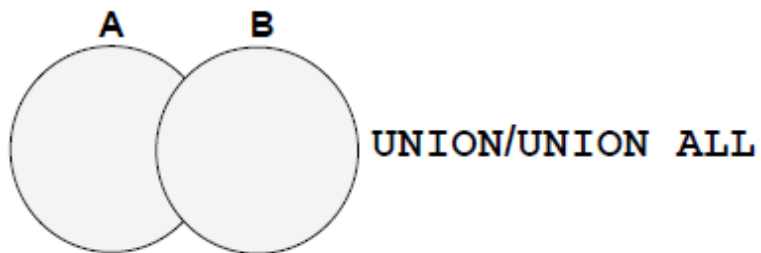
Correlated subqueries are used for row-by-row processing. Each subquery is executed once for every row of the outer query.



List employees whose salary is higher than their respective department's average salary

```
SELECT a.empno,a.deptno, a.sal  
FROM ap_emp a  
WHERE a.sal > (SELECT AVG(sal) deptavg FROM ap_emp b WHERE a.deptno=b.deptno);
```

The SET Operators



Operator	Returns
UNION	All distinct 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 that are selected by the first SELECT statement and that are not selected in the second SELECT statement

```

SELECT empno, ELNAME, deptno, job
FROM ap_emp
WHERE deptno in (20,30)
UNION
  SELECT empno, ELNAME, deptno, job
FROM ap_emp
WHERE job='ANALYST'
ORDER BY deptno, job;

```

```

SELECT empno, ELNAME, deptno, job
FROM ap_emp
WHERE deptno in (20,30)
UNION ALL
  SELECT empno, ELNAME, deptno, job
FROM ap_emp
WHERE job='ANALYST'
ORDER BY deptno, job;

```

```

SELECT empno, ELNAME, deptno, job
FROM ap_emp
WHERE deptno in (20,30)
INTERSECT
  SELECT empno, ELNAME, deptno, job
FROM ap_emp
WHERE job='ANALYST'
ORDER BY deptno, job;

```

```

SELECT empno, ELNAME, deptno, job
FROM ap_emp
WHERE deptno in (20,30)
MINUS
  SELECT empno, ELNAME, deptno, job
FROM ap_emp
WHERE job='ANALYST'
ORDER BY deptno, job;

```

What Is a View?

EMPLOYEES Table

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-97	AD_PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-98	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-98	AD_VP	17000
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000
							8000
							4200
							5800
							3600
							3100
							2500
							2300
							10500
							11000
							8800
							19000
							6000
							12000
							8300

20 rows selected.

ORACLE

11-4

Copyright © Oracle Corporation, 2001. All rights reserved.

What Is a View?

You can present logical subsets or combinations of data by creating views of tables. A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed. The tables on which a view is based are called base tables. The view is stored as a `SELECT` statement in the data dictionary.

Why Use Views?

- To restrict data access
- To make complex queries easy
- To provide data independence
- To present different views of the same data

== SIMPLE, SINGLE TABLE VIEW

```
create or replace view testview  
as  
select * from ap_emp  
where deptno=30  
;
```

```
SELECT * FROM testview;
```

```
update testview  
set sal=sal+100  
where empno=7499;
```

```
SELECT * FROM testview;
```

== READ ONLY VIEW

```
create or replace view testview  
as  
select * from ap_emp  
where deptno=30  
with READ ONLY;
```

```
SELECT * FROM testview;
```

```
update testview  
set sal=sal+100  
where empno=7499;
```

== VIEW WITH CHECK OPTION

```
create or replace view testview  
as  
select * from ap_emp  
where deptno=30  
with CHECK OPTION;
```

```
SELECT * FROM testview;
```

```
update testview  
set sal=sal+100  
where empno=7499;
```

```
UPDATE TESTVIEW  
SET DEPTNO=20  
WHERE EMPNO=7499;
```

== COMPLEX, MULTI TABLES / AGGREGATE functions

```

CREATE OR REPLACE VIEW dept30_V
AS
SELECT a.empno "Employee Number", a.hiredate "Hire Date", a. sal "Monthly Salary", b.dname "Department Name", b. loc
"Location"
FROM ap_emp a, ap_dept b
WHERE a.deptno=b.deptno AND
      a.deptno=30 ;

```

```

SELECT * FROM dept30_V;

```

```

SELECT * FROM USER_VIEWS
WHERE VIEW_NAME='DEPT30_V';

```

```

CREATE OR REPLACE VIEW emp_proj_detail_V
AS
SELECT a.empno AS Employee, a.ELNAME AS name, a.hiredate AS HireDate, a. sal AS Salary, b.dname AS Department, c.pname AS
Project, d.hours AS Hours
FROM ap_emp a, ap_dept b, ap_project c, ap_proemp d
WHERE a.deptno=b.deptno AND
      a.empno=d.empno AND
      d.projid=c.projid
ORDER BY 4;

```

```

SELECT * FROM emp_proj_detail_V;

```

```

SELECT name,sum(hours)
FROM ap_emp_proj_detail_V
GROUP BY name
ORDER by 2;

```

```
CREATE OR REPLACE VIEW dept_summary_V
(name,location,minsal,maxsal,avgsal)
AS
SELECT dname , loc, MIN(sal), MAX(sal) , AVG(sal)
FROM ap_emp a, ap_dept b
WHERE a.deptno=b.deptno
GROUP BY dname,loc;
```

```
SELECT * FROM dept_summary_V;
```

==== SUBQUERY in FROM CLAUSE (IN-LINE VIEWS)

- List employee number, name, department, salary along with their respective department's total number of employee, total salary, average salary, minimum, and maximum salary

```
SELECT a.empno, a.ELNAME, a.deptno,sal, b.depttempcnt , b.depttotsal , trunc(b.deptavgsal) deptavgsal ,
       b.deptminsal, b.deptmaxsal
FROM ap_emp a, ( select deptno, count(*) depttempcnt, sum(sal) depttotsal, avg(sal) deptavgsal, min(sal)
                deptminsal, max(sal) deptmaxsal from ap_emp group by deptno ) b
WHERE a.deptno=b.deptno ;
```

==== WITH CLAUSE [just like a temporary table that can be used multiple times in a query as needed]

WITH DEPTVAL AS

```
( select deptno, count(*) depttempcnt, sum(sal) depttotsal, avg(sal) deptavgsal, min(sal) deptminsal, max(sal) deptmaxsal
from ap_emp group by deptno)
SELECT a.empno, a.ELNAME, a.deptno,sal, b.depttempcnt , b.depttotsal , trunc(b.deptavgsal) deptavgsal, b.deptminsal,
b.deptmaxsal
FROM ap_emp a, DEPTVAL b
WHERE a.deptno=b.deptno ;
== CASE statement [Conditional processing]
```


Increase employee's salary by \$500 if their salary is more than \$5000, increase by \$300 if their salary is over \$3000, otherwise increase by \$100.

```
select sal, case
    when sal>5000 then sal+500
    when sal>3000 then sal+300
    else
        sal+100
    end "Salary Raise"
from ap_emp;
```

```
update ap_emp
set sal = case
    when sal>5000 then sal+500
    when sal>3000 then sal+300
    else
        sal+100
    end ;
```

-- Subquery in SELECT clause

-- List of department name, and total number of employees in those department

```
select dname name,
(select count(*)
from ap_emp B
where A.deptno= B.deptno)
as Num_Of_Employees
from ap_dept A ;
```

-- Subquery in HAVING clause

-- Department wise Total Salary for those departments which has total salary more than that of department 30

```
select deptno,sum(sal) Total_Salary  
from ap_emp  
group by deptno  
having sum(sal) >= (select sum(sal) from ap_emp where deptno=30);
```