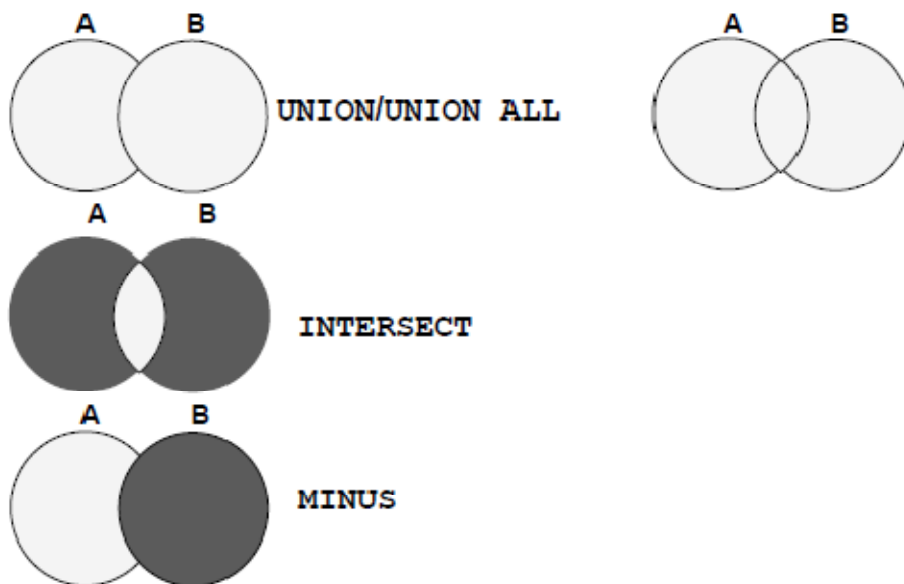


## Objectives

After completing this lesson, you should be able to do the following:

- Describe **SET** operators
  - Use a **SET** operator to combine multiple queries into a single query
  - Control the order of rows returned
- 

### The SET Operators



## The SET Operators

The SET operators combine the results of two or more component queries into one result. Queries containing SET operators are called *compound queries*.

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

All SET operators have equal precedence. If a SQL statement contains multiple SET operators, the Oracle server evaluates them from left (top) to right (bottom) if no parentheses explicitly specify another order. You should use parentheses to specify the order of evaluation explicitly in queries that use the INTERSECT operator with other SET operators.

**Note:** In the slide, the light color (grey) in the diagram represents the query result.

## Tables Used in This Lesson

The tables used in this lesson are:

- **EMPLOYEES:** Provides details regarding all current employees
- **JOB\_HISTORY:** When an employee switches jobs, the details of the start date and end date of the former job, the job identification number and department are recorded in this table

The structure and the data from the EMPLOYEES and the JOB\_HISTORY tables are shown on the next page.

There have been instances in the company of people who have held the same position more than once during their tenure with the company. For example, consider the employee Taylor, who joined the company on 24-MAR-1998. Taylor held the job title SA\_REP for the period 24-MAR-98 to 31-DEC-98 and the job title SA\_MAN for the period 01-JAN-99 to 31-DEC-99. Taylor moved back into the job title of SA\_REP, which is his current job title.

Similarly consider the employee Whalen, who joined the company on 17-SEP-1987. Whalen held the job title AD\_ASST for the period 17-SEP-87 to 17-JUN-93 and the job title AC\_ACCOUNT for the period 01-JUL-94 to 31-DEC-98. Taylor moved back into the job title of AD\_ASST, which is his current job title.

## Tables Used in This Lesson (continued)

DESC employees

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(5)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(5)
DEPARTMENT_ID		NUMBER(4)

```
SELECT employee_id, last_name, job_id, hire_date, department_id
FROM employees;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
100	King	AD_PRES	17-JUN-87	90
101	Kochhar	AD_VP	21-SEP-89	90
102	De Haan	AD_VP	13-JAN-93	90
103	Hunold	IT_PROG	03-JAN-90	60
104	Ernst	IT_PROG	21-MAY-91	60
107	Lorentz	IT_PROG	07-FEB-99	60
124	Mourgos	ST_MAN	16-NOV-99	50
141	Rajs	ST_CLERK	17-OCT-96	50
142	Davies	ST_CLERK	29-JAN-97	50
143	Matos	ST_CLERK	15-MAR-98	50
144	Vargas	ST_CLERK	09-JUL-98	50
149	Zlotkey	SA_MAN	29-JAN-00	80
174	Abel	SA_REP	11-MAY-96	80
176	Taylor	SA_REP	24-MAR-98	80
178	Grant	SA_REP	24-MAY-99	
200	Whalen	AD_ASST	17-SEP-87	10
201	Hartstein	MK_MAN	17-FEB-96	20
202	Fay	MK_REP	17-AUG-97	20
205	Higgins	AC_MGR	07-JUN-94	110
206	Gietz	AC_ACCOUNT	07-JUN-94	110

20 rows selected.

### Tables Used in This Lesson (continued)

DESC job\_history

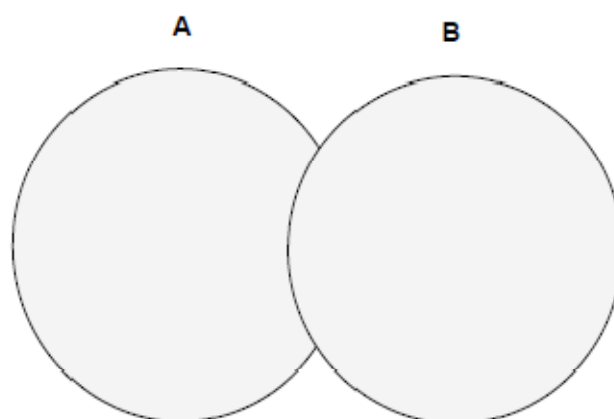
Name	Nul ?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

SELECT \* FROM job\_history;

EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
102	13-JAN-93	24-JUL-98	IT_PROG	60
101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
101	20-OCT-93	15-MAR-97	AC_MGR	110
201	17-FEB-96	18-DEC-99	MK_RFP	20
114	24-MAR-98	31-DEC-99	ST_CLERK	50
122	01-JAN-99	31-DEC-99	ST_CLERK	50
200	17-SEP-97	17-JUN-99	AD_ASST	90
176	24-MAR-98	31-DEC-98	SA_REP	80
176	01-JAN-99	31-DEC-99	SA_MAN	80
200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

10 rows selected.

## The UNION SET Operator



The **UNION** operator returns results from both queries after eliminating duplications.

## The UNION SET Operator

The UNION operator returns all rows selected by either query. Use the UNION operator to return all rows from multiple tables and eliminate any duplicate rows.

### Guidelines

- The number of columns and the data types of the columns being selected must be identical in all the SELECT statements used in the query. The names of the columns need not be identical.
- UNION operates over all of the columns being selected.
- NULL values are not ignored during duplicate checking.
- The IN operator has a higher precedence than the UNION operator.
- By default, the output is sorted in ascending order of the first column of the SELECT clause.

## Using the UNION Operator

Display the current and previous job details of all employees. Display each employee only once.

```
SELECT employee_id, job_id
FROM   employees
UNION
SELECT employee_id, job_id
FROM   job_history;
```

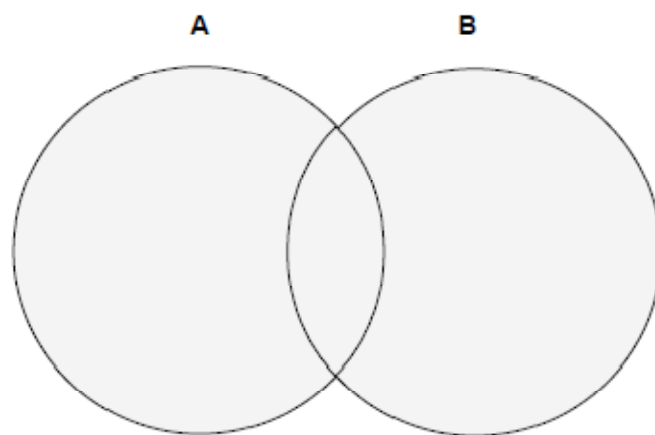
EMPLOYEE_ID	JOB_ID
100	AD_PRES
101	AC_ACCOUNT
101	AD_VP
178	SA_REP
200	AC_ACCOUNT
200	AD_ASST

28 rows selected.

Consider the following example:

```
SELECT employee_id, job_id, department_id
FROM   employees
UNION
SELECT employee_id, job_id, department_id
FROM   job_history;
```

## The UNION ALL Operator



The **UNION ALL** operator returns results from both queries including all duplications.

### The UNION ALL Operator

Use the **UNION ALL** operator to return all rows from multiple queries.

#### Guidelines

- Unlike **UNION**, duplicate rows are not eliminated and the output is not sorted by default.
- The **DISTINCT** keyword cannot be used.

**Note:** With the exception of the above, the guidelines for **UNION** and **UNION ALL** are the same.

## Using the UNION ALL Operator

Display the current and previous departments of all employees.

```
SELECT employee_id, job_id, department_id
FROM   employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM   job_history
ORDER BY employee_id;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
100	AD_PRES	90
174	SA_REP	80
176	SA_REP	80
176	SA_MAN	80
176	SA_REP	80
205	AC_MGR	110
206	AC_ACCOUNT	110

30 rows selected.

### The UNION ALL Operator (continued)

In the example, 30 rows are selected. The combination of the two tables totals to 30 rows. The UNION ALL operator does not eliminate duplicate records. The duplicate records are highlighted in the output shown in the slide. UNION returns all distinct rows selected by either query. UNION ALL returns all rows selected by either query, including all duplicates. Consider the query on the slide, now written with the UNION clause:

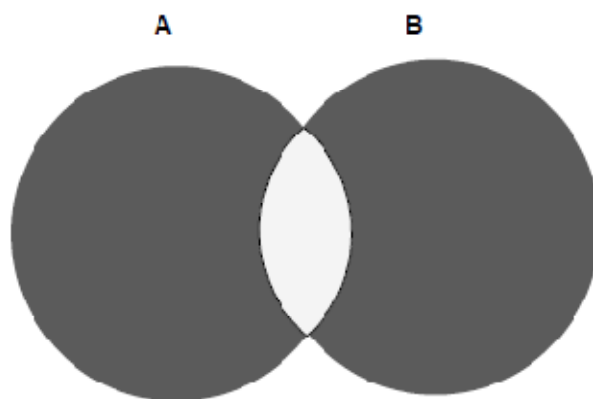
```
SELECT   employee_id, job_id, department_id
FROM     employees
UNION
SELECT   employee_id, job_id, department_id
FROM     job_history
ORDER BY employee_id;
```

The preceding query returns 29 rows. This is because it eliminates the following row (as it is a duplicate):

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
176	SA_REP	80



## The INTERSECT Operator



The **INTERSECT** operator returns results that are common to both queries.

### The INTERSECT Operator

Use the **INTERSECT** operator to return all rows common to multiple queries.

#### Guidelines

- The number of columns and the data types of the columns being selected by the **SELECT** statements in the queries must be identical in all the **SELECT** statements used in the query. The names of the columns need not be identical.
- Reversing the order of the intersected tables does not alter the result.
- **INTERSECT** does not ignore **NULL** values.

## Using the INTERSECT Operator

Display the employee IDs and job IDs of employees who are currently in a job title that they have held once before during their tenure with the company

```
SELECT employee_id, job_id
FROM   employees
INTERSECT
SELECT employee_id, job_id
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID
176	SA_REP
200	AD_ASST



In the example in this slide, the query returns only the records that have the same values in the selected columns in both tables.

What will be the results if you add the `DEPARTMENT_ID` column to the `SELECT` statement from the `EMPLOYEES` table and add the `DEPARTMENT_ID` column to the `SELECT` statement from the `JOB_HISTORY` table and run this query? The results may be different because of the introduction of another column whose values may or may not be duplicates.

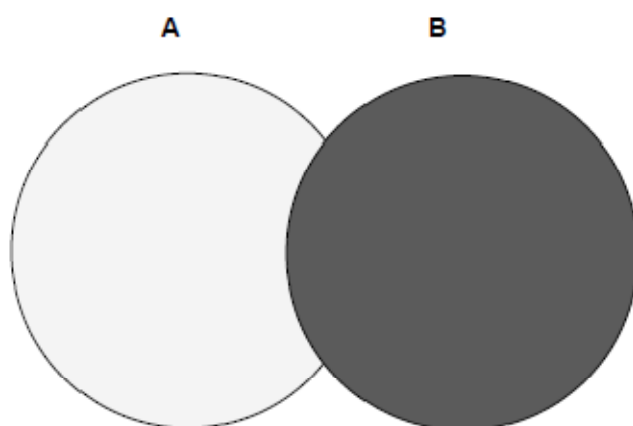
**Example**

```
SELECT employee_id, job_id, department_id
FROM   employees
INTERSECT
SELECT employee_id, job_id, department_id
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
176	SA_REP	80

Employee 200 is no longer part of the results because the `EMPLOYEES.DEPARTMENT_ID` value is different from the `JOB_HISTORY.DEPARTMENT_ID` value.

## The MINUS Operator



The **MINUS** operator returns rows from the first query that are not present in the second query.

## The MINUS Operator

Use the MINUS operator to return rows returned by the first query that are not present in the second query (the first SELECT statement MINUS the second SELECT statement).

### Guidelines

- The number of columns and the data types of the columns being selected by the SELECT statements in the queries must be identical in all the SELECT statements used in the query. The names of the columns need not be identical.
- All of the columns in the WHERE clause must be in the SELECT clause for the MINUS operator to work.

## The MINUS Operator

Display the employee IDs of those employees who have not changed their jobs even once.

```
SELECT employee_id
FROM employees
MINUS
SELECT employee_id
FROM job_history;
```

EMPLOYEE_ID	JOB_ID
100	AD_PRES
101	AD_VP
102	AD_ASST
103	SA_MAN
104	SA_MAN
105	SA_MAN
106	SA_MAN
107	SA_MAN
108	SA_MAN
109	SA_MAN
110	SA_MAN
111	SA_MAN
112	SA_MAN
113	SA_MAN
114	SA_MAN
115	SA_MAN
116	SA_MAN
117	SA_MAN
118	SA_MAN
119	SA_MAN
120	SA_MAN
121	MK_MAN
122	MK_REP
123	MK_REP
124	MK_REP
125	MK_REP
126	MK_REP
127	MK_REP
128	MK_REP
129	MK_REP
130	MK_REP
131	MK_REP
132	MK_REP
133	MK_REP
134	MK_REP
135	MK_REP
136	MK_REP
137	MK_REP
138	MK_REP
139	MK_REP
140	MK_REP
141	MK_REP
142	MK_REP
143	MK_REP
144	MK_REP
145	MK_REP
146	MK_REP
147	MK_REP
148	MK_REP
149	MK_REP
150	MK_REP
151	MK_REP
152	MK_REP
153	MK_REP
154	MK_REP
155	MK_REP
156	MK_REP
157	MK_REP
158	MK_REP
159	MK_REP
160	MK_REP
161	MK_REP
162	MK_REP
163	MK_REP
164	MK_REP
165	MK_REP
166	MK_REP
167	MK_REP
168	MK_REP
169	MK_REP
170	MK_REP
171	MK_REP
172	MK_REP
173	MK_REP
174	MK_REP
175	MK_REP
176	MK_REP
177	MK_REP
178	MK_REP
179	MK_REP
180	MK_REP
181	MK_REP
182	MK_REP
183	MK_REP
184	MK_REP
185	MK_REP
186	MK_REP
187	MK_REP
188	MK_REP
189	MK_REP
190	MK_REP
191	MK_REP
192	MK_REP
193	MK_REP
194	MK_REP
195	MK_REP
196	MK_REP
197	MK_REP
198	MK_REP
199	MK_REP
200	MK_REP
201	MK_REP
202	MK_REP
203	MK_REP
204	MK_REP
205	MK_REP
206	MK_REP
207	MK_REP
208	MK_REP
209	MK_REP
210	MK_REP
211	MK_REP
212	MK_REP
213	MK_REP
214	MK_REP
215	MK_REP
216	MK_REP
217	MK_REP
218	MK_REP
219	MK_REP
220	MK_REP
221	MK_REP
222	MK_REP
223	MK_REP
224	MK_REP
225	MK_REP
226	MK_REP
227	MK_REP
228	MK_REP
229	MK_REP
230	MK_REP
231	MK_REP
232	MK_REP
233	MK_REP
234	MK_REP
235	MK_REP
236	MK_REP
237	MK_REP
238	MK_REP
239	MK_REP
240	MK_REP
241	MK_REP
242	MK_REP
243	MK_REP
244	MK_REP
245	MK_REP
246	MK_REP
247	MK_REP
248	MK_REP
249	MK_REP
250	MK_REP
251	MK_REP
252	MK_REP
253	MK_REP
254	MK_REP
255	MK_REP
256	MK_REP
257	MK_REP
258	MK_REP
259	MK_REP
260	MK_REP
261	MK_REP
262	MK_REP
263	MK_REP
264	MK_REP
265	MK_REP
266	MK_REP
267	MK_REP
268	MK_REP
269	MK_REP
270	MK_REP
271	MK_REP
272	MK_REP
273	MK_REP
274	MK_REP
275	MK_REP
276	MK_REP
277	MK_REP
278	MK_REP
279	MK_REP
280	MK_REP
281	MK_REP
282	MK_REP
283	MK_REP
284	MK_REP
285	MK_REP
286	MK_REP
287	MK_REP
288	MK_REP
289	MK_REP
290	MK_REP
291	MK_REP
292	MK_REP
293	MK_REP
294	MK_REP
295	MK_REP
296	MK_REP
297	MK_REP
298	MK_REP
299	MK_REP
300	MK_REP
301	MK_REP
302	MK_REP
303	MK_REP
304	MK_REP
305	MK_REP
306	MK_REP
307	MK_REP
308	MK_REP
309	MK_REP
310	MK_REP
311	MK_REP
312	MK_REP
313	MK_REP
314	MK_REP
315	MK_REP
316	MK_REP
317	MK_REP
318	MK_REP
319	MK_REP
320	MK_REP
321	MK_REP
322	MK_REP
323	MK_REP
324	MK_REP
325	MK_REP
326	MK_REP
327	MK_REP
328	MK_REP
329	MK_REP
330	MK_REP
331	MK_REP
332	MK_REP
333	MK_REP
334	MK_REP
335	MK_REP
336	MK_REP
337	MK_REP
338	MK_REP
339	MK_REP
340	MK_REP
341	MK_REP
342	MK_REP
343	MK_REP
344	MK_REP
345	MK_REP
346	MK_REP
347	MK_REP
348	MK_REP
349	MK_REP
350	MK_REP
351	MK_REP
352	MK_REP
353	MK_REP
354	MK_REP
355	MK_REP
356	MK_REP
357	MK_REP
358	MK_REP
359	MK_REP
360	MK_REP
361	MK_REP
362	MK_REP
363	MK_REP
364	MK_REP
365	MK_REP
366	MK_REP
367	MK_REP
368	MK_REP
369	MK_REP
370	MK_REP
371	MK_REP
372	MK_REP
373	MK_REP
374	MK_REP
375	MK_REP
376	MK_REP
377	MK_REP
378	MK_REP
379	MK_REP
380	MK_REP
381	MK_REP
382	MK_REP
383	MK_REP
384	MK_REP
385	MK_REP
386	MK_REP
387	MK_REP
388	MK_REP
389	MK_REP
390	MK_REP
391	MK_REP
392	MK_REP
393	MK_REP
394	MK_REP
395	MK_REP
396	MK_REP
397	MK_REP
398	MK_REP
399	MK_REP
400	MK_REP
401	MK_REP
402	MK_REP
403	MK_REP
404	MK_REP
405	MK_REP
406	MK_REP
407	MK_REP
408	MK_REP
409	MK_REP
410	MK_REP
411	MK_REP
412	MK_REP
413	MK_REP
414	MK_REP
415	MK_REP
416	MK_REP
417	MK_REP
418	MK_REP
419	MK_REP
420	MK_REP
421	MK_REP
422	MK_REP
423	MK_REP
424	MK_REP
425	MK_REP
426	MK_REP
427	MK_REP
428	MK_REP
429	MK_REP
430	MK_REP
431	MK_REP
432	MK_REP
433	MK_REP
434	MK_REP
435	MK_REP
436	MK_REP
437	MK_REP
438	MK_REP
439	MK_REP
440	MK_REP
441	MK_REP
442	MK_REP
443	MK_REP
444	MK_REP
445	MK_REP
446	MK_REP
447	MK_REP
448	MK_REP
449	MK_REP
450	MK_REP
451	MK_REP
452	MK_REP
453	MK_REP
454	MK_REP
455	MK_REP
456	MK_REP
457	MK_REP
458	MK_REP
459	MK_REP
460	MK_REP
461	MK_REP
462	MK_REP
463	MK_REP
464	MK_REP
465	MK_REP
466	MK_REP
467	MK_REP
468	MK_REP
469	MK_REP
470	MK_REP
471	MK_REP
472	MK_REP
473	MK_REP
474	MK_REP
475	MK_REP
476	MK_REP
477	MK_REP
478	MK_REP
479	MK_REP
480	MK_REP
481	MK_REP
482	MK_REP
483	MK_REP
484	MK_REP
485	MK_REP
486	MK_REP
487	MK_REP
488	MK_REP
489	MK_REP
490	MK_REP
491	MK_REP
492	MK_REP
493	MK_REP
494	MK_REP
495	MK_REP
496	MK_REP
497	MK_REP
498	MK_REP
499	MK_REP
500	MK_REP
501	MK_REP
502	MK_REP
503	MK_REP
504	MK_REP
505	MK_REP
506	MK_REP
507	MK_REP
508	MK_REP
509	MK_REP
510	MK_REP
511	MK_REP
512	MK_REP
513	MK_REP
514	MK_REP
515	MK_REP
516	MK_REP
517	MK_REP
518	MK_REP
519	MK_REP
520	MK_REP
521	MK_REP
522	MK_REP
523	MK_REP
524	MK_REP
525	MK_REP
526	MK_REP
527	MK_REP
528	MK_REP
529	MK_REP
530	MK_REP
531	MK_REP
532	MK_REP
533	MK_REP
534	MK_REP
535	MK_REP
536	MK_REP
537	MK_REP
538	MK_REP
539	MK_REP
540	MK_REP
541	MK_REP
542	MK_REP
543	MK_REP
544	MK_REP
545	MK_REP
546	MK_REP
547	MK_REP
548	MK_REP
549	MK_REP
550	MK_REP
551	MK_REP
552	MK_REP
553	MK_REP
554	MK_REP
555	MK_REP
556	MK_REP
557	MK_REP
558	MK_REP
559	MK_REP
560	MK_REP
561	MK_REP
562	MK_REP
563	MK_REP
564	MK_REP
565	MK_REP
566	MK_REP
567	MK_REP
568	MK_REP
569	MK_REP
570	MK_REP
571	MK_REP
572	MK_REP
573	MK_REP
574	MK_REP
575	MK_REP
576	MK_REP
577	MK_REP
578	MK_REP
579	MK_REP
580	MK_REP
581	MK_REP
582	MK_REP
583	MK_REP
584	MK_REP
585	MK_REP
586	MK_REP
587	MK_REP
588	MK_REP
589	MK_REP
590	MK_REP
591	MK_REP
592	MK_REP
593	MK_REP
594	MK_REP
595	MK_REP
596	MK_REP
597	MK_REP
598	MK_REP
599	MK_REP
600	MK_REP
601	MK_REP
602	MK_REP
603	MK_REP
604	MK_REP
605	MK_REP
606	MK_REP
607	MK_REP
608	MK_REP
609	MK_REP
610	MK_REP
611	MK_REP
612	MK_REP
613	MK_REP
614	MK_REP
615	MK_REP
616	MK_REP
617	MK_REP
618	MK_REP
619	MK_REP
620	MK_REP
621	MK_REP
622	MK_REP
623	MK_REP
624	MK_REP
625	MK_REP
626	MK_REP
627	MK_REP
628	MK_REP
629	MK_REP
630	MK_REP
631	MK_REP
632	MK_REP
633	MK_REP
634	MK_REP
635	MK_REP
636	MK_REP
637	MK_REP
638	MK_REP
639	MK_REP
640	MK_REP
641	MK_REP
642	MK_REP
643	MK_REP
644	MK_REP
645	MK_REP
646	MK_REP
647	MK_REP
648	MK_REP
649	MK_REP
650	MK_REP
651	MK_REP
652	MK_REP
653	MK_REP
654	MK_REP
655	MK_REP
656	MK_REP
657	MK_REP
658	MK_REP
659	MK_REP
660	MK_REP
661	MK_REP
662	MK_REP
663	MK_REP
664	MK_REP
665	MK_REP
666	MK_REP
667	MK_REP
668	MK_REP
669	MK_REP
670	MK_REP
671	MK_REP
672	MK_REP
673	MK_REP
674	MK_REP
675	MK_REP
676	MK_REP
677	MK_REP
678	MK_REP
679	MK_REP
680	MK_REP
681	MK_REP
682	MK_REP
683	MK_REP
684	MK_REP
685	MK_REP
686	MK_REP
687	MK_REP
688	MK_REP
689	MK_REP
690	MK_REP
691	MK_REP
692	MK_REP
693	MK_REP
694	MK_REP
695	MK_REP
696	MK_REP
697	MK_REP
698	MK_REP
699	MK_REP
700	MK_REP
701	MK_REP
702	MK_REP
703	MK_REP
704	MK_REP
705	MK_REP
706	MK_REP
707	MK_REP
708	MK_REP
709	MK_REP
710	MK_REP
711	MK_REP
712	MK_REP
713	MK_REP
714	MK_REP
710	

## SET Operator Guidelines

- The expressions in the **SELECT** lists must match in number and data type.
- Parentheses can be used to alter the sequence of execution.
- The **ORDER BY** clause:
  - Can appear only at the very end of the statement
  - Will accept the column name, aliases from the first **SELECT** statement, or the positional notation

### SET Operator Guidelines

- The expressions in the select lists of the queries must match in number and datatype. Queries that use **UNION**, **UNION ALL**, **INTERSECT**, and **MINUS** SET operators in their **WHERE** clause must have the same number and type of columns in their **SELECT** list. For example:

```
SELECT employee_id, department_id
FROM   employees
WHERE  (employee_id, department_id)
      IN (SELECT employee_id, department_id
          FROM   employees
          UNION
          SELECT  employee_id, department_id
          FROM    job_history);
```
- The **ORDER BY** clause:
  - Can appear only at the very end of the statement
  - Will accept the column name, an alias, or the positional notation
- The column name or alias, if used in an **ORDER BY** clause, must be from the first **SELECT** list.
- **SET** operators can be used in subqueries.

## The Oracle Server and SET Operators

- Duplicate rows are automatically eliminated except in UNION ALL.
- Column names from the first query appear in the result.
- The output is sorted in ascending order by default except in UNION ALL.

When a query uses SET operators, the Oracle Server eliminates duplicate rows automatically except in the case of the UNION ALL operator. The column names in the output are decided by the column list in the first SELECT statement. By default, the output is sorted in ascending order of the first column of the SELECT clause.

The corresponding expressions in the select lists of the component queries of a compound query must match in number and datatype. If component queries select character data, the data type of the return values are determined as follows:

- If both queries select values of datatype CHAR, the returned values have datatype CHAR.
- If either or both of the queries select values of datatype VARCHAR2, the returned values have datatype VARCHAR2.

## Matching the SELECT Statements

Using the UNION operator, display the department ID, location, and hire date for all employees.

```
SELECT department_id, TO_NUMBER(null) location, hire_date
FROM   employees
UNION
SELECT department_id, location_id, TO_DATE(null)
FROM   departments;
```

DEPARTMENT_ID	LOCATION	HIRE_DATE
10	1700	
10		17-SEP-87
20	1800	
20		17-FEB-96

190	1700	24-MAY-98
-----	------	-----------

27 rows selected.

As the expressions in the select lists of the queries must match in number, you can use dummy columns and the data type conversion functions to comply with this rule. In the slide, the name location is given as the dummy column heading. The TO\_NUMBER function is used in the first query to match the NUMBER data type of the LOCATION\_ID column retrieved by the second query. Similarly, the TO\_DATE function in the second query is used to match the DATE datatype of the HIRE\_DATE column retrieved by the second query.

## Matching the SELECT Statement

Using the UNION operator, display the employee ID, job ID, and salary of all employees.

```
SELECT employee_id, job_id, salary
FROM   employees
UNION
SELECT employee_id, job_id, 0
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID	SALARY
100	AD_PRES	24000
101	AC_ACCOUNT	0
101	AC_MGR	0
205	AC_MGR	12000
206	AC_ACCOUNT	8300

30 rows selected.

The EMPLOYEES and JOB\_HISTORY tables have several columns in common; for example, EMPLOYEE\_ID, JOB\_ID and DEPARTMENT\_ID. But what if you want the query to display the EMPLOYEE\_ID, JOB\_ID, and SALARY using the UNION operator, knowing that the salary exists only in the EMPLOYEES table?

The code example in the slide matches the EMPLOYEE\_ID and the JOB\_ID columns in the EMPLOYEES and in the JOB\_HISTORY tables. A literal value of 0 is added to the JOB\_HISTORY SELECT statement to match the numeric SALARY column in the EMPLOYEES SELECT statement.

In the preceding results, each row in the output that corresponds to a record from the JOB\_HISTORY table contains a 0 in the SALARY column.

## Controlling the Order of Rows

Produce an English sentence using two UNION operators.

```
COLUMN a_dummy NOPRINT
SELECT 'sing' AS "My dream", 3 a_dummy
FROM dual
UNION
SELECT 'I'd like to teach', 1
FROM dual
UNION
SELECT 'the world to', 2
FROM dual
ORDER BY 2;
```

My dream
I'd like to teach
the world to
sing

By default, the output is sorted in ascending order on the first column. You can use the ORDER BY clause to change this.

### Using ORDER BY to Order Rows

The ORDER BY clause can be used only once in a compound query. If used, the ORDER BY clause must be placed at the end of the query. The ORDER BY clause accepts the column name, an alias, or the positional notation. Without the ORDER BY clause, the code example in the slide produces the following output in the alphabetical order of the first column:

My dream
I'd like to teach
sing
the world to

**Note:** Consider a compound query where the UNION SET operator is used more than once. In this case, the ORDER BY clause can use only positions rather than explicit expressions.

## ASSIGNMENTS

1. List the department IDs for departments that do not contain the job ID ST\_CLERK, using SET operators.

DEPARTMENT_ID
10
20
60
80
90
110
190

7 rows selected.

2. Display the country ID and the name of the countries that have no departments located in them, using SET operators.

CO	COUNTRY_NAME
DE	Germany

3. Produce a list of jobs for departments 10, 50, and 20, in that order. Display job ID and department ID, using SET operators.

JOB_ID	DEPARTMENT_ID
AD_ASST	10
ST_CLERK	50
ST_MAN	50
MK_MAN	20
MK_REP	20

4. List the employee IDs and job IDs of those employees who are currently in the job title that they have held once before during their tenure with the company.

EMPLOYEE_ID	JOB_ID
176	SA_REP
200	AD_ASST



5. Write a compound query that lists the following:

- Last names and department ID of all the employees from the EMPLOYEES table, irrespective of the fact whether they belong to any department or not
- Department ID and department name of all the departments from the DEPARTMENTS table, irrespective of the fact whether they have employees working in them or not.

LAST_NAME	DEPARTMENT_ID	TO_CHAR(NULL)
Abel	80	
Davies	50	
De Haan	90	
Ernst	60	
Fay	20	
Gietz	110	
Grant		
Hartstein	20	
Higgins	110	
Hunold	60	
King	90	
Kochhar	90	
Lorentz	60	
Matos	50	
Mourgos	50	
Rajs	50	
Taylor	80	
Vargas	50	
Whalen	10	
Zlotkey	80	
	10	Administration
	20	Marketing
	50	Shipping
	60	IT
	80	Sales
	90	Executive
	110	Accounting
	190	Contracting

28 rows selected.