

SQL: Advanced Queries

- Chapter 7 in Murach text
- Subqueries
 - Subqueries in WHERE, HAVING, FROM and SELECT clauses
 - Correlated subqueries
 - Subqueries in INSERT, UPDATE and DELETE statements (see Chapter 5)

SQL: Subqueries

Four ways to introduce a subquery in a SELECT statement

1. In a WHERE clause as a search condition
2. In a HAVING clause as a search condition
3. In the FROM clause as a table specification
4. In the SELECT clause as a column specification

Subquery in WHERE Clause

4	•	SELECT AVG(invoice_total)
5		FROM invoices;
6		

Result Grid			Filter Rows:	
AVG(invoice_total)				
1879.741316				

228		
229	•	SELECT invoice_number, invoice_date, invoice_total
230		FROM invoices
231		WHERE invoice_total > (SELECT AVG(invoice_total) FROM invoices)
232		ORDER BY invoice_total;
233		

Result Grid						Filter Rows:		Export:		Wrap Cell Content:	
	invoice_number	invoice_date	invoice_total								
	989319-487	2018-06-20	1927.54								
	97/522	2018-06-28	1962.13								
	989319-417	2018-07-23	2051.59								
	989319-427	2018-06-16	2115.81								
	989319-477	2018-06-08	2184.11								
	587056	2018-06-30	2184.50								
	989319-497	2018-06-12	2312.20								
	989319-467	2018-07-01	2318.03								
	367447	2018-06-11	2433.00								
	989319-437	2018-06-01	2765.36								

** When using a subquery with a comparison operator in a WHERE clause, the subquery should return a single value.

Subquery in WHERE Clause

```
19 • SELECT vendor_id
20 FROM vendors
21 WHERE vendor state = "CA";
```

Result Grid
vendor_id
4
6
7
8
9
10
11
12
13
14
15
16

```
239 • SELECT invoice_number, invoice_date, invoice_total
240 FROM invoices
241 WHERE vendor_id IN (SELECT vendor_id
242 FROM vendors
243 WHERE vendor_state="CA")
244 ORDER BY invoice_date, invoice_number;
245
246
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
invoice_number	invoice_date	invoice_total	
125520-1	2018-04-24	95.00	
97/488	2018-04-24	601.95	
111-92R-10096	2018-04-30	16.33	
25022117	2018-05-01	6.00	
P02-88D77S7	2018-05-03	856.92	
OP58872	2018-05-07	116.54	
24863706	2018-05-10	6.00	
10843	2018-05-11	4901.26	

** When using a subquery with an IN operator in a WHERE clause, the subquery should return a single column of values.

Subquery vs Join

```
239 • SELECT invoice_number, invoice_date, invoice_total
240 FROM invoices
241 WHERE vendor_id IN (SELECT vendor_id
242 FROM vendors
243 WHERE vendor_state="CA")
244 ORDER BY invoice_date, invoice_number;
245
246
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
invoice_number	invoice_date	invoice_total	
125520-1	2018-04-24	95.00	
97/488	2018-04-24	601.95	
111-92R-10096	2018-04-30	16.33	
25022117	2018-05-01	6.00	
P02-88D77S7	2018-05-03	856.92	
OP58872	2018-05-07	116.54	
24863706	2018-05-10	6.00	
10843	2018-05-11	4901.26	

```
245
246 • SELECT invoice_number, invoice_date, invoice_total
247 FROM invoices JOIN vendors
248 ON invoices.vendor_id = vendors.vendor_id
249 WHERE vendor_state="CA"
250 ORDER BY invoice_date, invoice_number;
251
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
invoice_number	invoice_date	invoice_total	
125520-1	2018-04-24	95.00	
97/488	2018-04-24	601.95	
111-92R-10096	2018-04-30	16.33	
25022117	2018-05-01	6.00	
P02-88D77S7	2018-05-03	856.92	
OP58872	2018-05-07	116.54	
24863706	2018-05-10	6.00	
10843	2018-05-11	4901.26	
77290	2018-05-13	1750.00	
121897	2018-05-19	450.00	

Subquery vs Join

Advantages of joins

- A join can include columns from both tables.
- A join is more intuitive when it uses an existing relationship.

Advantages of subqueries

- A subquery can pass an aggregate value to the main query.
- A subquery is more intuitive when it uses an ad hoc relationship.
- Long, complex queries can be easier to code using subqueries.

Subquery vs Join

```
35 • SELECT v.vendor_id, vendor_state, vendor_name
36 FROM vendors v LEFT JOIN invoices i
37     ON v.vendor_id = i.vendor_id
38 WHERE i.vendor_id IS NULL
39 ORDER BY v.vendor_id;
```

Result Grid |   Filter Rows: | Export:  | Wrap Cells

vendor_id	vendor_state	vendor_name
1	WI	US Postal Service
2	DC	National Information Data Ctr
3	DC	Register of Copyrights
4	CA	Jobtrak
5	NJ	Newbridge Book Clubs
6	CA	California Chamber Of Commerce
7	CA	Towne Advertiser's Mailing Svcs
8	CA	BFI Industries
9	CA	Pacific Gas & Electric
10	CA	Robbins Mobile Lock And Key

```
41 • SELECT vendor_id, vendor_state, vendor_name
42 FROM vendors
43 WHERE vendor_id NOT IN
44     (SELECT vendor_id FROM invoices)
45 ORDER BY vendor_id;
```

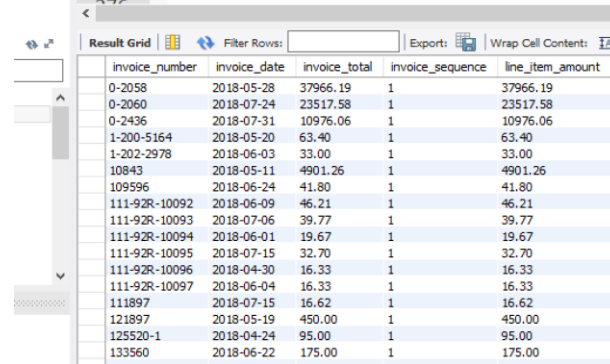
Result Grid |   Filter Rows: | Edit:   

vendor_id	vendor_state	vendor_name
1	WI	US Postal Service
2	DC	National Information Data Ctr
3	DC	Register of Copyrights
4	CA	Jobtrak
5	NJ	Newbridge Book Clubs
6	CA	California Chamber Of Commerce
7	CA	Towne Advertiser's Mailing Svcs
8	CA	BFI Industries
9	CA	Pacific Gas & Electric
10	CA	Robbins Mobile Lock And Key

Correlated Subqueries

Query that returns invoices
with line items

```
268  
269 -- correlated subqueries  
270 • SELECT invoice_number, invoice_date, invoice_total, invoice_sequence,  
271 line_item_amount  
272 FROM invoices JOIN invoice_line_items  
273 ON invoices.invoice_id = invoice_line_items.invoice_id  
274 ORDER BY invoice_number, invoice_sequence;  
275  
276
```

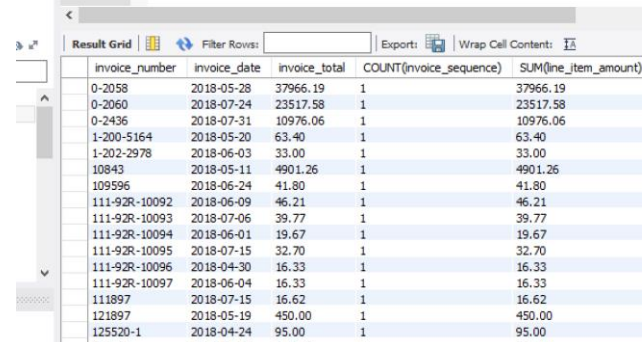


The screenshot shows a database query result grid with the following columns: invoice_number, invoice_date, invoice_total, invoice_sequence, and line_item_amount. The data is sorted by invoice_number and then by invoice_sequence. The grid displays 20 rows of data, including invoices 0-2058, 0-2060, 0-2436, 1-200-5164, 1-202-2978, 10843, 109596, 111-92R-10092, 111-92R-10093, 111-92R-10094, 111-92R-10095, 111-92R-10096, 111-92R-10097, 111897, 121897, 125520-1, and 133560.

invoice_number	invoice_date	invoice_total	invoice_sequence	line_item_amount
0-2058	2018-05-28	37966.19	1	37966.19
0-2060	2018-07-24	23517.58	1	23517.58
0-2436	2018-07-31	10976.06	1	10976.06
1-200-5164	2018-05-20	63.40	1	63.40
1-202-2978	2018-06-03	33.00	1	33.00
10843	2018-05-11	4901.26	1	4901.26
109596	2018-06-24	41.80	1	41.80
111-92R-10092	2018-06-09	46.21	1	46.21
111-92R-10093	2018-07-06	39.77	1	39.77
111-92R-10094	2018-06-01	19.67	1	19.67
111-92R-10095	2018-07-15	32.70	1	32.70
111-92R-10096	2018-04-30	16.33	1	16.33
111-92R-10097	2018-06-04	16.33	1	16.33
111897	2018-07-15	16.62	1	16.62
121897	2018-05-19	450.00	1	450.00
125520-1	2018-04-24	95.00	1	95.00
133560	2018-06-22	175.00	1	175.00

Query that returns invoices
with count and sum of all
line items

```
275  
276 • SELECT invoice_number, invoice_date, invoice_total, COUNT(invoice_sequence),  
277 SUM(line_item_amount)  
278 FROM invoices JOIN invoice_line_items  
279 ON invoices.invoice_id = invoice_line_items.invoice_id  
280 GROUP BY invoice_number  
281 ORDER BY invoice_number, invoice_sequence;  
282  
283
```



The screenshot shows a database query result grid with the following columns: invoice_number, invoice_date, invoice_total, COUNT(invoice_sequence), and SUM(line_item_amount). The data is sorted by invoice_number and then by invoice_sequence. The grid displays 20 rows of data, including invoices 0-2058, 0-2060, 0-2436, 1-200-5164, 1-202-2978, 10843, 109596, 111-92R-10092, 111-92R-10093, 111-92R-10094, 111-92R-10095, 111-92R-10096, 111-92R-10097, 111897, 121897, 125520-1, and 133560.

invoice_number	invoice_date	invoice_total	COUNT(invoice_sequence)	SUM(line_item_amount)
0-2058	2018-05-28	37966.19	1	37966.19
0-2060	2018-07-24	23517.58	1	23517.58
0-2436	2018-07-31	10976.06	1	10976.06
1-200-5164	2018-05-20	63.40	1	63.40
1-202-2978	2018-06-03	33.00	1	33.00
10843	2018-05-11	4901.26	1	4901.26
109596	2018-06-24	41.80	1	41.80
111-92R-10092	2018-06-09	46.21	1	46.21
111-92R-10093	2018-07-06	39.77	1	39.77
111-92R-10094	2018-06-01	19.67	1	19.67
111-92R-10095	2018-07-15	32.70	1	32.70
111-92R-10096	2018-04-30	16.33	1	16.33
111-92R-10097	2018-06-04	16.33	1	16.33
111897	2018-07-15	16.62	1	16.62
121897	2018-05-19	450.00	1	450.00
125520-1	2018-04-24	95.00	1	95.00
133560	2018-06-22	175.00	1	175.00

Correlated Subqueries

Insert invoice 99999-999
with incorrect invoice_total
field.

```
284
285 • INSERT INTO invoices VALUES
286 (115,122,'99999-99','2018-05-28','5000.00','5000.00','0.00',3,'2018-06-30','218-06-15');
287
288 • INSERT INTO invoice_line_items VALUES
289 (115,1,553,'5000.00','Freight'),
290 (115,2,570,'99.99','Office Max');
```

invoice_number	invoice_date	invoice_total	COUNT(invoice_sequence)	SUM(line_item_amount)
97/503	2018-05-30	639.77	1	639.77
97/522	2018-06-28	1962.13	2	1962.13
97/553	2018-06-25	904.14	1	904.14
97/553B	2018-06-10	313.55	1	313.55
972110	2018-05-15	207.78	1	207.78
989319-417	2018-07-23	2051.59	1	2051.59
989319-427	2018-06-16	2115.81	1	2115.81
989319-437	2018-06-01	2765.36	1	2765.36
989319-447	2018-07-24	3689.99	1	3689.99
989319-457	2018-04-08	3813.33	1	3813.33
989319-467	2018-07-01	2318.03	1	2318.03
989319-477	2018-06-08	2184.11	1	2184.11
989319-487	2018-06-20	1927.54	1	1927.54
989319-497	2018-06-12	2312.20	1	2312.20
9982771	2018-07-24	503.20	1	503.20
99999-99	2018-05-28	5000.00	2	5099.99
C73-24	2018-07-19	600.00	1	600.00

Correlated Subqueries

Query checks each invoice and filters for invoices whose invoice_total not equal to sum of invoice line_item_amounts

```
292 • SELECT invoice_number, invoice_date, invoice_total
293 FROM invoices
294 WHERE invoice_total <>
295     (SELECT SUM(line_item_amount)
296      FROM invoice_line_items
297      WHERE invoices.invoice_id = invoice_line_items.invoice_id);
```

Result Grid | | Filter Rows: | Export: | Wrap Cell Content:


	invoice_number	invoice_date	invoice_total
1	99999-99	2018-05-28	5000.00

Correlated subqueries run once for every row of the main query (similar to a for loop in other languages).

Correlated Subqueries

Correlated subquery with
self-join

```
80 • SELECT vendor_id, invoice_number, invoice_total
81 FROM invoices i
82 WHERE invoice_total >
83     (SELECT AVG(invoice_total)
84      FROM invoices
85      WHERE vendor_id = i.vendor_id)
86 ORDER BY vendor_id, invoice_total;
87
88
```

Result Grid |   Filter Rows: | Export:  | Wrap Cell Conte

vendor_id	invoice_number	invoice_total
34	0545443	1083.58
37	547480102	224.00
37	547481328	224.00
72	40318	21842.00
80	133560	175.00
83	31359783	1575.00

Subquery in SELECT Clause

```
310
311 • SELECT vendor_name,
312       (SELECT MAX(invoice_date)
313        FROM invoices
314        WHERE vendor_id=vendors.vendor_id) AS latest_invoice
315 FROM vendors
316 ORDER BY latest_invoice DESC;
317
318
```

Result Grid

vendor_name	latest_invoice
Federal Express Corporation	2018-08-02
Blue Cross	2018-08-01
Mallov Lithoagraphing Inc.	2018-07-31
Cardinal Business Media, Inc.	2018-07-28
Zvlka Design	2018-07-25
Ford Motor Credit Companv	2018-07-24
United Parcel Service	2018-07-24
Ingram	2018-07-21
Wakefield Co	2018-07-20
Reiter's Scientific & Pro Books	2018-07-19
Pacific Bell	2018-07-15
Suburban Propane	2018-07-15

Subquery in SELECT statement

```
317
318 • SELECT vendor_name, MAX(invoice_date) AS latest_invoice
319       FROM vendors LEFT JOIN invoices
320       ON vendors.vendor_id=invoices.vendor_id
321 GROUP BY vendor_name
322 ORDER BY latest_invoice DESC;
323
```

Result Grid

vendor_name	latest_invoice
Federal Express Corporation	2018-08-02
Blue Cross	2018-08-01
Mallov Lithoagraphing Inc.	2018-07-31
Cardinal Business Media, Inc.	2018-07-28
Zvlka Design	2018-07-25
Ford Motor Credit Companv	2018-07-24
United Parcel Service	2018-07-24
Ingram	2018-07-21

JOIN instead

Subquery in FROM Clause

Query that returns sum of invoices by vendor state and name

```
107 • SELECT vendor_state, vendor_name, SUM(invoice_total) AS sum_of_invoices
108 FROM vendors JOIN invoices
109 ON vendors.vendor_id = invoices.vendor_id
110 GROUP BY vendor_state, vendor_name
111 ORDER BY vendor_state, sum_of_invoices DESC;
112
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

vendor_state	vendor_name	sum_of_invoices
AZ	Wells Fargo Bank	662.00
CA	Digital Dreamworks	7125.34
CA	Zvika Design	6940.25
CA	Bertelsmann Industrv Svcs. Inc	6940.25
CA	Yesmed. Inc	4901.26
CA	Computerworld	2433.00

Query that selects max invoice total by state from the query results above

```
114 • SELECT vendor_state, MAX(sum_of_invoices) AS max_invoice_sum
115 FROM (
116     SELECT vendor_state, SUM(invoice_total) AS sum_of_invoices
117     FROM vendors JOIN invoices
118     ON vendors.vendor_id = invoices.vendor_id
119     GROUP BY vendor_state, vendor_name
120 ) AS vendors_by_state
121 GROUP BY vendor_state
122 ORDER BY vendor_state;
123
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

vendor_state	max_invoice_sum
AZ	662.00
CA	7125.34
DC	600.00
MA	1367.50
MI	119892.41
NV	23177.96

** When using a subquery in FROM Clause, subquery returns data frame with rows and columns (called an “inline view”)

Query 1 Results						Query 2 Results	
vendor_state	vendor_name	sum_of_invoices				vendor_state	max_invoice_sum
AZ	Wells Fargo Bank	662				AZ	662
CA	Digital Dreamworks	7125.34				CA	7125.34
CA	Zylka Design	6940.25				DC	600
CA	Bertelsmann Industry Svcs. Inc	6940.25				MA	1367.5
CA	Yesmed, Inc	4901.26				MI	119892.41
CA	Computerworld	2433				NV	28177.96
CA	Pollstar	1750				OH	207.78
CA	Franchise Tax Board	1600				PA	265.36
CA	IBM	1200.12				TN	4378.02
CA	Wang Laboratories, Inc.	936.93				TX	2154.42
CA	Fresno County Tax Collector	856.92					
CA	Blue Cross	564					
CA	Ford Motor Credit Company	503.2					
CA	Gostanian General Building	450					
CA	Wakefield Co	356.48					
CA	Postmaster	290					
CA	Dristas Groom & McCormick	220					
CA	Pacific Bell	171.01					
CA	Evans Executone Inc	95					
CA	Roadway Package System, Inc	43.67					
CA	Coffee Break Service	41.8					
CA	Abbey Office Furnishings	17.5					
CA	Suburban Propane	16.62					
DC	Reiter's Scientific & Pro Books	600					
MA	Dean Witter Reynolds	1367.5					
MI	Malloy Lithographing Inc	119892.41					
MI	Data Reproductions Corp	21927.31					
NV	United Parcel Service	28177.96					
NV	Cahners Publishing Company	2184.5					
OH	Edward Data Services	207.78					
OH	Compuserve	19.9					
PA	Cardinal Business Media, Inc.	265.36					
TN	Federal Express Corporation	4378.02					
TX	Ingram	2154.42					