

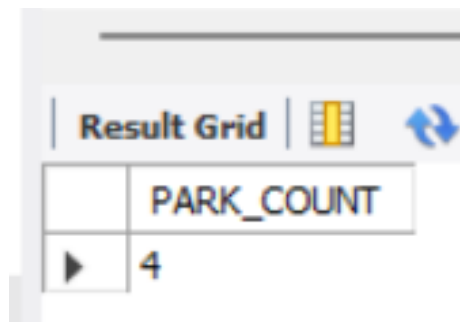
NUR ALEEYA ALEESYA BINTI HEZRIE

2022892198

CDCS1103C

1. Write a query to calculate the number of unique park code that exist in the TICKET table, replace calculated column with 'Park_Count'.

```
234 • SELECT COUNT(DISTINCT(PARK_CODE)) AS 'PARK_COUNT'
235 FROM TICKET;
236
```

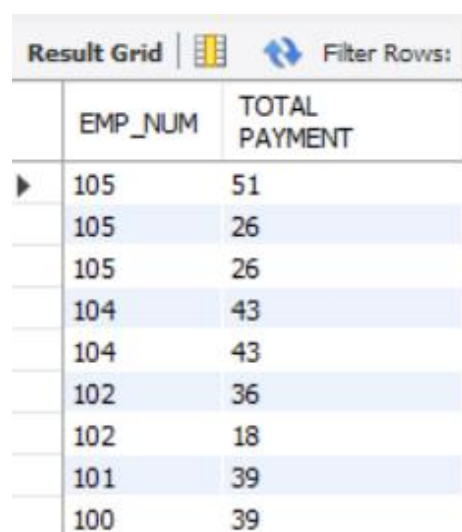


The screenshot shows a 'Result Grid' window with a single column header 'PARK_COUNT' and one row containing the value '4'.

PARK_COUNT
4

2. Display the employee numbers of all employees and the total payment (Hour rate multiply Hour per attract) they have worked. Use 'Total payment' for calculated column. Sort the output according to employee number descending.

```
9 • SELECT EMP_NUM, ROUND(HOUR_RATE * HOURS_PER_ATTRACT) AS 'TOTAL PAYMENT'
0 FROM HOURS
1 ORDER BY EMP_NUM DESC;
```



The screenshot shows a 'Result Grid' window with two columns: 'EMP_NUM' and 'TOTAL PAYMENT'. The results are sorted by 'EMP_NUM' in descending order.

EMP_NUM	TOTAL PAYMENT
105	51
105	26
105	26
104	43
104	43
102	36
102	18
101	39
100	39

3. Display the employee numbers of all employees and the total hours they have worked. Use 'Total Hours' for calculated column. Sort the output according employee number descending.

```
243 • SELECT EMP_NUM,  
244 SUM(HOURS_PER_ATTRACT) AS 'TOTAL HOURS'  
245 FROM HOURS  
246 GROUP BY EMP_NUM  
247 ORDER BY EMP_NUM DESC;
```

Result Grid			Filter Rows:
	EMP_NUM	TOTAL HOURS	
▶	105	12	
	104	12	
	102	9	
	101	6	
	100	6	

4. Show the attraction number and the minimum and maximum hourly rate for each attraction. Use alias MIN and MAX for calculated column.

```
251 • SELECT ATTRACT_NO, MIN(HOUR_RATE) AS 'MIN',  
252 MAX(HOUR_RATE) AS 'MAX'  
253 FROM HOURS  
254 GROUP BY ATTRACT_NO;
```

Result Grid				Filter Rows:
	ATTRACT_NO	MIN	MAX	
▶	10034	6.50	6.50	
	10078	8.50	8.50	
	10098	8.50	8.50	
	30011	7.20	7.20	
	30012	5.99	8.50	
	30044	5.99	5.99	

5. Write a query to show the transaction numbers and AVERAGE line prices (use the SALES_LINE table). Display average value that are greater than €50 only.

```
258 • SELECT TRANSACTION_NO,  
259     AVG(LINE_PRICE) AS 'AVERAGE'  
260 FROM SALES_LINE  
261 GROUP BY TRANSACTION_NO  
262 HAVING AVG(LINE_PRICE) > 50;
```

Result Grid			Filter Rows:
	TRANSACTION_NO	AVERAGE	
▶	12782	69.980000	
	12785	63.313333	
	34534	70.960000	
	34535	84.200000	
	34537	53.350000	
	34539	53.090000	
	34540	70.960000	
	34541	84.200000	
	67592	114.680000	

6. Display Transaction Number, Sale date information from the SALES table. Calculate the number of sales after 1 January 2007. Sort the result in descending order of the sale date.

```
264 • SELECT TRANSACTION_NO, SALE_DATE,  
265 COUNT(*) AS 'NUMBER OF SALES'  
266 FROM SALES  
267 WHERE SALE_DATE > '2007-01-01'  
268 GROUP BY TRANSACTION_NO, SALE_DATE  
269 ORDER BY SALE_DATE DESC;
```

Result Grid				Filter Rows:	Export:
	TRANSACTION_NO	SALE_DATE	NUMBER OF SALES		
▶	12781	2007-05-18	1		
	12782	2007-05-18	1		
	12783	2007-05-18	1		
	12784	2007-05-18	1		
	12785	2007-05-18	1		
	12786	2007-05-18	1		
	34534	2007-05-18	1		
	34535	2007-05-18	1		
	34536	2007-05-18	1		
	34537	2007-05-18	1		
	34538	2007-05-18	1		
	34539	2007-05-18	1		
	34540	2007-05-18	1		
	34541	2007-05-18	1		
	67589	2007-05-18	1		
	67590	2007-05-18	1		
Result 52					
	67591	2007-05-18	1		
	67592	2007-05-18	1		
	67593	2007-05-18	1		

7. Using the TICKET table, write a query to display the park code and the average ticket price. Limiting the average ticket price greater or equal to 20. Replace calculated column name using 'AVERAGE PRICE'.

```
271 • SELECT PARK_CODE,  
272     AVG(TICKET_PRICE) AS 'AVERAGE PRICE'  
273 FROM TICKET  
274 GROUP BY PARK_CODE  
275 HAVING AVG(TICKET_PRICE) >= 20;
```

Result Grid			Filter Rows:	
	PARK_CODE	AVERAGE PRICE		
▶	FR1001	24.990000		
	UK3452	25.196667		