IT2351 / IT2851 / IT2552 / IT2152 / IT2651

Database Management Systems

Structured Query Language (B)
Advanced SELECT

Topics (Advanced SELECT)

- Using Aggregate functions
- Using GROUP BY
- Using HAVING
- Using Subqueries

Aggregate / Group Functions

- Aggregate (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	2600
50	2500
80	10500
80	11000
80	8600
	7000
10	4400

...

20 rows selected.

Maximum salary in EMPLOYEES table

MAX(SALARY)
24000

Use of Aggregate Functions

- 5 aggregate functions can be applied on column values :
  - COUNT, SUM, AVG, MIN, MAX
  - Each operates on a single column of a table and **return single value**
    - Select SUM(total_price) from order_detail ;
  - Use in SELECT, HAVING clauses only

ISO standard defines five aggregate functions:

Function	Format	Returns
COUNT	**COUNT(*)**	**Counts all rows of a table, regardless of whether nulls or duplicate values occur.**
	COUNT(DISTINCT column-name)	**No. of unique values in the specified column.**
AVG	**AVG (column-name)**	**Average of values in a specified numeric column.**
SUM	**SUM (column-name)**	**Sum of values in a specified numeric column.**
MIN	**MIN (column-name)**	**Lowest value in the specified column.**
MAX	**MAX (column-name)**	**Highest value in the specified column.**

Use of Aggregate Functions

- COUNT
  - Find the total number of orders received
    - Select COUNT(*) from orders ;

ORDER_NUM	ORDER_DATE	CUSTOMER _NUM	SHIP_DATE
1001	20-May-06	103	1-Jun-06
1002	21-May-06	101	26-May-06
1003	22-May-06	103	23-May-06
1004	22-May-06	102	30-May-06

Orders

Count(*)
4

1 row selected

Use of Aggregate Functions

- COUNT(DISTINCT)
  - Find the total number of customers who have placed orders
    - Select COUNT(DISTINCT customer_num)
      from orders ;

ORDER_NUM	ORDER_DATE	CUSTOMER_NUM	SHIP_DATE
1001	20-May-06	103	1-Jun-06
1002	21-May-06	101	26-May-06
1003	22-May-06	103	23-May-06
1004	22-May-06	102	30-May-06

Orders

Count(*)
3

1 row selected

- What would be the result if the keyword DISTINCT is not used ?
- What is the difference between count customer from orders table and from customer table?

Use of Aggregate Functions

- AVG( )
  - Find the average product unit price
    - Select AVG(unit_price) from product ;

Product

PROD_NUM	UNIT_PRICE
113	681
120	37
130	

Avg(unit_price)
359

1 row selected

359? 239.33? Null?

359

The null value will not be included for compute average

Use of Aggregate Functions

- SUM( )
  - Find the total sales from all the orders
    - Select SUM(total_price) from order_detail ;

ORDER_NUM	TOTAL_PRICE
1001	100
1001	250
1002	100

Order_detail

Sum(total_price)
450

1 row selected

Use of Aggregate Functions

- MIN( ), MAX( )
  - Find the maximum and the minimum shipment charge
    - Select MAX(ship_charge), MIN(ship_charge)
      from orders ;

ORDER_NUM	SHIP_CHARGE
1001	150
1001	250
1002	75

Order_detail

Max(ship_charge)	min(ship_charge)
250	75

1 row selected

GROUP BY *column list* clause

- To group rows and produce sub-totals
  - i.e. to produce a single summary row for each group
- *Column list* :
  - any base table columns
  - calculated columns
- How it works :
  - Partitions a table into groups of rows that have *something in common* to apply a function on each group of rows.
  - Most often *combined with aggregate functions* that produce summary values for each of those groups

GROUP BY *column list* clause

- Example 1 : Find the number of items in each order

order_num	item_num	total_price
1001	1	$250.00
1002	1	$960.00
1007	2	$126.00
1007	5	$600.00
1002	2	$240.00
1007	1	$250.00
1008	1	$840.00
1007	3	$240.00
1007	4	$480.00
..		
..		
1008	2	$100.00

Order 1001 has 1 item

Order 1002 has 2 items

Order 1007 has 5 items

Order 1008 has 2 items

Order_Detail

This kind of operation requires the **Group by** clause.
In this example, we are grouping by order_num.

Group by order_num

GROUP BY *column list* clause

- Example 1 : Find the number of items in each order
  - Partition ORDER_DETAIL table by ORDER_NUM to count number of items per order.

order_num	item_num	total_price	count(*)
1001	1	$250.00	count=1
1002	1	$960.00	count=2
1002	2	$240.00	
..			
..			
1007	1	$250.00	count=5
1007	2	$126.00	
1007	3	$240.00	
1007	4	$480.00	
1007	5	$600.00	
1008	1	$840.00	count=2
1008	2	$100.00	

```
Select    order_num, count(*)
From      order_detail
Group by order_num
```

One line will be printed for each group

GROUP BY *column list* clause

- Select order_num, count(*)
  from order_detail
  group by order_num ;

- Example 1 (output) :

order_num	**count(*)**
1001	1
1002	2
…..	….
1007	5
1008	2
.....	....

GROUP BY *column list* clause

- Example 2 : Find the total price of each order
  - Partition ORDER_DETAIL table by ORDER_NUM to find out the sum of the item's total price in each order.

order_num	total_price	...	sum(total_price)
1001	$250.00		$250.00
1002	$960.00		
1002	$240.00		$1,200.00
..			
..			
1007	$250.00		
1007	$126.00		
1007	$240.00		$1,696.00
1007	$480.00		
1007	$600.00		
1008	$840.00		
1008	$100.00		$940.00

```
Select      order_num,
            sum(total_price)
from        order_detail
group by order_num ;
```

GROUP BY *column list* clause

- Select order_num, sum(total_price)
  from order_detail
  group by order_num ;

- Example 2 (output) :

  - **order_num** **sum(total_price)**

1001	250
1002	1200
…	…
1007	1696
1008	940

GROUP BY *column list* clause

- Example 3 (GROUP BY used when joining tables) :
  - List number of order items for all the orders made by customer with customer_num = 110

```
Select      o.order_num, count(*)
from        orders o
inner join  order_detail d on o.order_num = d.order_num
where       o.customer_num = 110
group by    o.order_num ;
```

  - **Output :**

order_num	count (*)
1008	2
1015	1

GROUP BY - Additional Notes

- **All column names in** SELECT **list must appear in** GROUP BY **clause unless name is used only in an aggregate function, e.g.**
  - SELECT **a, b**, sum(c)
    FROM t1
    GROUP BY **a, b** … correct
  - SELECT **x, y**, avg(z)
    FROM t2
    GROUP BY **x** … incorrect
- **If** WHERE **is used with** GROUP BY**,** WHERE **is applied first, then groups are formed from remaining rows satisfying predicate.**

HAVING *condition* clause

- **To <u>restrict the groups</u> in the final result**

- **Compare with** WHERE **which filters individual rows :**
  - The condition in "HAVING clause" always includes at least one aggregate function, otherwise the search condition could be moved to the WHERE clause and applied to individual rows.

  - Thus **aggregate functions cannot be used in the WHERE clause**.

HAVING *condition* clause

- Example 1 : List orders with more than 2 items
  - Select order_num, count(*)
    from order_detail
    group by order_num
    having count(*) > 2 ;
  - Partition ORDER_DETAIL table by ORDER_NUM, show orders with more than 2 items.

ORDER_NUM	TOTAL_PRICE	…	COUNT(*)	COUNT(*) > 2
1001	$250.00		= 1	***NO***
1002	$960.00		= 2	***NO***
1002	$240.00			
..				
..				
1007	$250.00		= 5	***YES***
1007	$126.00			
1007	$240.00			
1007	$480.00			
1007	$600.00			
1008	$840.00		= 2	***NO***
1008	$100.00			

HAVING *condition* clause

- Example 1 (output) :
  - **order_num** **count(*)**
    1007 5

Review

- Study the SQLs below, explain the differences. How many rows do you expect to find in the output list ?
  - Select avg(unit_price)
    from product;
  - Select avg(unit_price)
    from product
    where prod_num = '101';
  - Select prod_num, avg(unit_price)
    from product
    group by prod_num;

Questions for query formulation

- SELECT ..
  - What are the output ? Base table/derived columns, aggregate values ?
- FROM …
  - Where to get the output from ? Which table(s) ?
  - Also include tables of which the columns are needed for specifying the WHERE conditions
- INNER JOIN.. ON
  - Include other tables needed in joining. One for each additional table.
  - What are the common columns in each table?
- WHERE …
  - Specify the filter conditions on individual records
- GROUP BY ..
  - How do you want to partition the records in the table ?
- HAVING ..
  - How do you want to select the groups ?
- ORDER BY ..
  - How do you want the output to be sorted ?

Subqueries in SQL

- Subqueries
  - Some SQL statements can have a **SELECT** embedded within them.
  - This embedded SELECT statement is called **Subquery**
  - A subquery can be used in **WHERE** and **HAVING** clauses of an outer **SELECT**
  - Subqueries may also appear in **INSERT**, **UPDATE**, and **DELETE**s.

Subqueries in SQL

- Subquery with Equality
  - List the most expensive product.
    - Select prod_num
      from product
      where **unit_price** =(select **max(unit_price)**
      from product);

Subqueries in SQL

- Subquery with Equality
  - List customers who reside in 'California'

```
Select    fname, lname
from      customer
where     state_code = (select  state_code
                        from    state
                        where state_name = 'California');
```

  - Note that this query can also be written using a join :

```
Select    fname, lname
from      customer c
inner join state s
on        c.state_code = s.state_code
where     state_name = 'California';
```

Subqueries in SQL

- Subquery with Aggregate
  - List those employees with salary higher than the average salary of all employees
    - Select emp_id
from employee
where salary > (select AVG(salary) from employee);
  - Cannot write 'WHERE salary > AVG(salary)'. (recall : **cannot use aggregate functions in WHERE clause**)
  - Instead, use subquery to find the average salary, and then use outer SELECT to find those employees with salary greater than that.

Summary

- Advanced SELECT statement
  - Includes **GROUP BY** and **HAVING** clause in the basic statement.
  - Creates subqueries.

- The syntax to include all the clauses in the SELECT statement is:

SELECT [**DISTINCT**] *select list*
FROM *table list*
{[**INNER JOIN** *tablename* **ON** *condition*]}
[**WHERE** *condition*]
[**GROUP BY** *column list*]
[**HAVING** *condition*]
[**ORDER BY** *column list* [**DESC**]]

QUIZ

- Give an example of using GROUP BY clause.
- When do we need to use the HAVING clause?
- Which clause shall be used if the result displayed is arrange in descending?