

5.4

Improving the GROUP BY Query

Improving the GROUP BY Query

- The report would be nicer if we showed the category name instead of the category_id. This will require joining the product table to the category table.
- We can **ROUND** the **AVG** list price by category to TWO decimals points.
- We can **CONCAT** the dollar sign to the left of the list_price.

Code Sample:

```
USE bike;
SELECT category_name,
       CONCAT('$', ROUND(AVG(list_price),2)) AS 'Average List Price'
FROM product p
     JOIN category c
       ON p.category_id = c.category_id
GROUP BY category_name
ORDER BY category_name;
```

Output:

The screenshot shows a SQL IDE interface. At the top, there's a toolbar with various icons and a text 'Limit to 1000 rows'. Below the toolbar, a SQL query is entered in a text area:

```

1 • USE bike;
2 • SELECT category_name,
3       CONCAT('$', ROUND(AVG(list_price),2)) AS 'Average List Price'
4 FROM product p
5     JOIN category c
6     ON p.category_id = c.category_id
7 GROUP BY category_name
8 ORDER BY category_name;

```

Below the query editor, there's a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table:

category_name	Average List Price
Children Bicycles	\$287.79
Comfort Bicycles	\$682.12
Cruisers Bicycles	\$730.41
Cyclocross Bicycles	\$2542.79
Electric Bikes	\$3281.66
Mountain Bikes	\$1649.76
Road Bikes	\$3175.36

USE bike:

- Set the bike database to be the default

SELECT category_name,

CONCAT('\$', ROUND(AVG(list_price),2)) AS 'Average List Price'

- Return the category_name from the category table.
- You do not have to qualify the column name with the table name because category_name only exists in one table of the join.
- Return the list price with the '\$' followed by the list_price rounded to the 2nd decimal and assigned a column alias of 'Average List Price'.
- You do not have to qualify the column name of list_price because it exists in only one table of the join.

FROM product p

JOIN category c

ON p.category_id = c.category_id

- JOIN the product table to the category table
- Assign a table alias of "p" to product and "c" to category
- The join condition is the primary key of category_id from the category table equal to the foreign key of category_id in the product table.

GROUP BY category_name

- Instead of retrieving a single value with the average price of all products, return a list of average prices by category name.

ORDER BY category_name;

- Sort the results by category_name



This content is provided to you freely by BYU-I Books.

Access it online or download it at https://books.byui.edu/learning_mysql/improving_the_group_.

