

Project Portfolio

Jacob Huckleberry

ITS410: Database Management

Colorado State University Global

Dr. Raquel Hicks

August 7, 2022

Figure 1

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

1 SELECT
2   product_name,
3   list_price,
4   discount_percent,
5   ROUND(list_price * (discount_percent * .01), 2) AS discount_amount,
6   ROUND(list_price - (list_price * (discount_percent * .01)), 2) AS discount_price
7 FROM
8   products
9 ORDER BY discount_price DESC
10 LIMIT 5;
11

```

The results pane shows the following data:

product_name	list_price	discount_percent	discount_amount	discount_price
Gibson SG	2517.00	52.00	1308.84	1208.16
Gibson Les Paul	1199.00	30.00	359.70	839.30
Yamaha DGX 640 88-Key Digital Piano	799.99	0.00	0.00	799.99
Tama 5-Piece Drum Set with Cymbals	789.99	15.00	120.00	679.99
Fender Precision	759.99	30.00	240.00	559.99

The Action Output pane shows the following details:

	Time	Action	Response	Duration / Fetch Time
1	14:51:18	SELECT product_name, list_price, discount_percent, ROUND(list_price * (dis...	5 row(s) returned	0.00073 sec / 0.0000...

Query Completed

Figure 1 shows a query which selects the product name, list price and discount percent columns from the products table. The query also creates a column called `discount_amount` which is calculated from the discount percent and list price columns. The query creates another column called `discount_price` which takes the same calculation from the `discount_amount` column but subtracts it with the `list_price` value. Both of these created columns are rounded to the nearest hundredth. The query is in descending order by the discount price column and is limited to return the first five rows.

Figure 2

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

1  SELECT
2      item_id,
3      item_price,
4      discount_amount,
5      quantity,
6      item_price * quantity AS price_total,
7      discount_amount * quantity AS discount_total,
8      (item_price - discount_amount) * quantity AS item_total
9  FROM
10     order_items
11  HAVING item_total > 500
12  ORDER BY item_total DESC;

```

The Results grid shows the following data:

item_id	item_price	discount_amount	quantity	price_total	discount_total	item_total
5	1199.00	369.70	2	2398.00	739.40	1658.60
3	2517.00	1308.84	1	2517.00	1308.84	1208.16
1	1199.00	369.70	1	1199.00	369.70	829.30
11	799.99	120.00	1	799.99	120.00	679.99
9	799.99	240.00	1	799.99	240.00	559.99

The Action Output pane shows the following log:

Time	Action	Response	Duration / Fetch Time
14:51:18	SELECT product_name, list_price, discount_percent, RO...	5 row(s) returned	0.00073 sec / 0.0000...
15:07:30	SELECT item_id, item_price, discount_amount, quantity...	Error Code: 1054, Unknown column 'item_total' in 'where clause'	0.00035 sec
15:15:30	SELECT item_id, item_price, discount_amount, quantity...	5 row(s) returned	0.00090 sec / 0.000...
15:18:06	SELECT item_id, item_price, discount_amount, quantity...	5 row(s) returned	0.00064 sec / 0.000...

Figure 2 shows a query which selects the item id, item price, discount amount and quantity columns from the order_items table. The query also creates three columns called price_total, discount_total and item_total. The price_total is calculated by the product of the item price and quantity. The discount total is calculated from the product of the item price and quantity. The item total is calculated from the subtraction of the item price and discount amount all multiplied by the quantity. Lastly, the query is in descending order by the item total and only displays results which are above the item total of 500.

Figure 3

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

1 SELECT DISTINCT
2   p1.product_name, p1.list_price
3 FROM
4   products p1
5   JOIN
6   products p2 ON p1.product_id <> p2.product_id
7   AND p1.list_price = p2.list_price
8 ORDER BY product_name;

```

The Results tab shows the output of the query:

product_name	list_price
Fender Precision	799.99
Tama 5-Piece Drum Set with Cymbals	799.99
Yamaha DGX 640 88-Key Digital Piano	799.99

The Action Output tab shows the execution log:

Time	Action	Response	Duration / Fetch Time
15:07:30	SELECT item_id, item_price, discount_amount, quantity...	Error Code: 1054, Unknown column 'item_total' in 'where clause'	0.00035 sec
15:15:30	SELECT item_id, item_price, discount_amount, quantity...	5 row(s) returned	0.00090 sec / 0.000...
15:18:06	SELECT item_id, item_price, discount_amount, quantity...	5 row(s) returned	0.00064 sec / 0.000...
15:32:13	SELECT p1.product_name, p1.list_price FROM products p1 JOI...	6 row(s) returned	0.00067 sec / 0.0000...
15:32:50	SELECT DISTINCT p1.product_name, p1.list_price FROM produ...	3 row(s) returned	0.00065 sec / 0.000...

Figure 3 shows a query which uses a self join to return the distinct rows of the product name and the list price columns. The query uses aliasing and comparison operators to complete the self join and is ordered by the product name column.

Figure 4

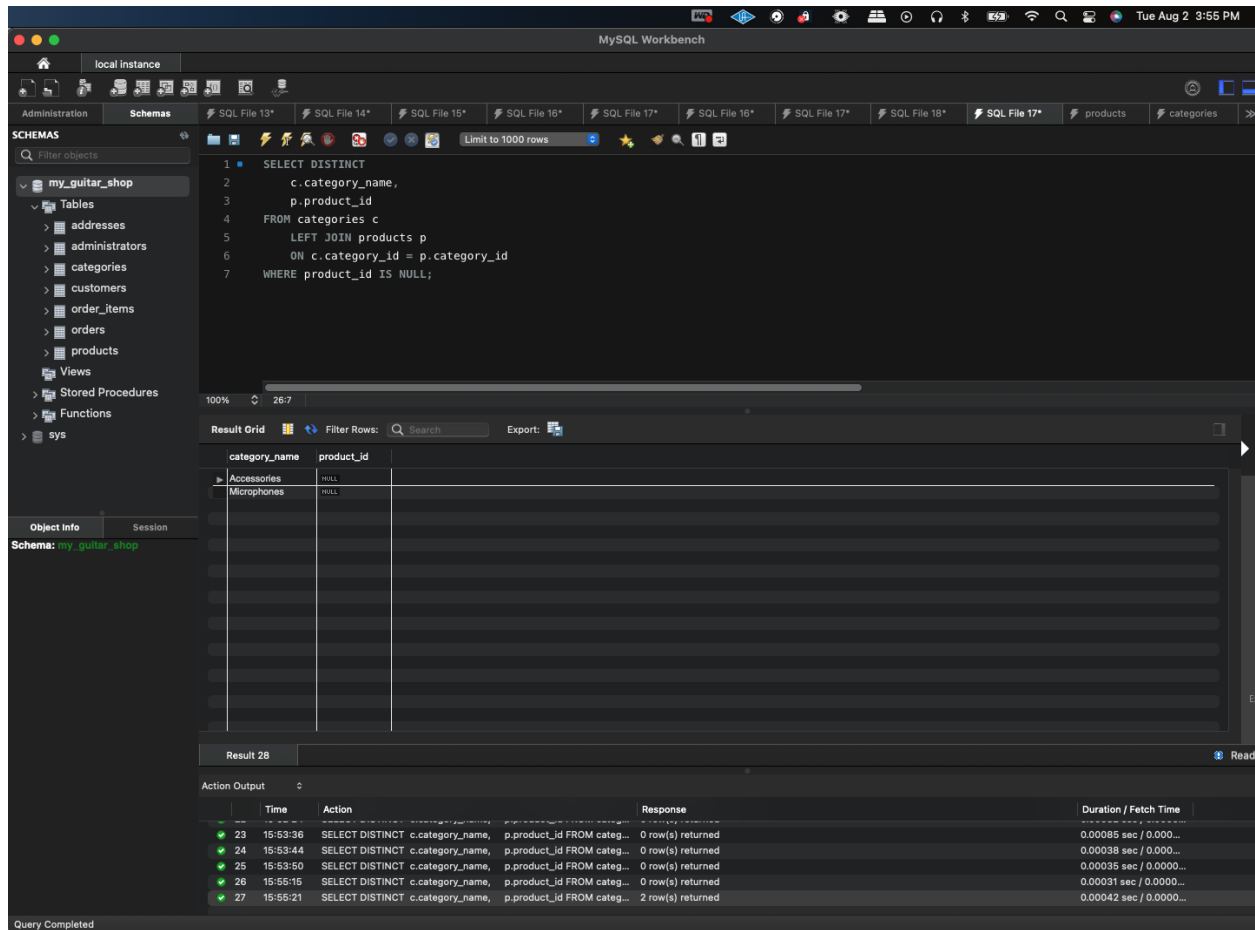


Figure 4 shows a query which selects each row of the category name and the product id column which has not been used. This is completed by an outer join on the category id column with the search condition of product id being a null value.

Figure 5

The screenshot shows the MySQL Workbench interface. The top toolbar includes icons for Administration, Schemas, and various SQL file operations. The left sidebar displays the 'SCHEMAS' tree with 'my_guitar_shop' expanded, showing tables like 'addresses', 'administrators', 'categories', and 'customers'. The 'customers' table is selected, and its columns are listed: customer_id, email_address, password, first_name, last_name, shipping_address_id, and billing_address_id. The main editor shows a SQL query:

```
1 INSERT INTO customers (
2   customer_id, email_address, password, first_name, last_name
3 )
4 VALUES (
5   DEFAULT, 'rick@raven.com', '', 'Rick', 'Raven'
6 );
7
8 SELECT * FROM my_guitar_shop.customers;
```

The 'Result Grid' at the bottom displays the results of the query. It shows a table with 7 columns: customer_id, email_address, password, first_name, last_name, shipping_address_id, and billing_address_id. The data includes 10 rows, with the last row being 'Rick Raven' with a null shipping and billing address. The 'Action Output' pane at the bottom shows the execution log:

Time	Action	Response	Duration / Fetch Time
27 16:55:21	SELECT DISTINCT c.category_name, p.product_id FROM category...	2 row(s) returned	0.00042 sec / 0.0000...
28 16:00:26	SELECT * FROM my_guitar_shop.customers LIMIT 0, 1000	8 row(s) returned	0.0050 sec / 0.00001...
29 16:03:42	INSERT INTO customers (customer_id, email_address, password,...	1 row(s) affected	0.0034 sec
30 16:03:48	SELECT * FROM my_guitar_shop.customers LIMIT 0, 1000	9 row(s) returned	0.00044 sec / 0.000...
31 16:04:35	SELECT * FROM my_guitar_shop.customers LIMIT 0, 1000	9 row(s) returned	0.00038 sec / 0.000...

Figure 5 shows a query which inserts a row into the customers table. This insert operator defaults the customer id, has specific values for the next four columns and leaves the shipping address id and billing address id values null.

Figure 6

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

1 SELECT
2   c.email_address, COUNT(o.customer_id) AS product_count
3 FROM
4   customers c
5   JOIN
6     orders o ON c.customer_id = o.customer_id
7 GROUP BY c.email_address
8 HAVING product_count > 1;

```

The query results are displayed in a table with two columns: `email_address` and `product_count`. The results are as follows:

email_address	product_count
allan.sherwood@yahoo.com	2
david.goldstein@hotmail.com	2

The bottom panel shows the Action Output with the following entries:

Time	Action	Response	Duration / Fetch Time
16:04:35	SELECT * FROM my_guitar_shop.customers LIMIT 0, 1000	9 row(s) returned	0.00038 sec / 0.000...
16:08:39	SELECT * FROM my_guitar_shop.orders LIMIT 0, 1000	9 row(s) returned	0.0024 sec / 0.00001...
16:12:30	SELECT c.email_address, COUNT(o.customer_id) FROM customer...	Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list conta...	0.00039 sec
16:13:10	SELECT c.email_address, COUNT(o.customer_id) FROM cust...	7 row(s) returned	0.00099 sec / 0.000...
16:14:05	SELECT c.email_address, COUNT(o.customer_id) AS product_c...	2 row(s) returned	0.00050 sec / 0.0000...

Figure 6 shows a query which selects the email address column and the count of the customer id as a column named `product_count`. This query joins the tables and sorts them by the email address and only displays the rows of the customers who have more than one product purchase.

Figure 7

The screenshot shows the MySQL Workbench interface. The top toolbar includes icons for Administration, Schemas, and various SQL file operations. The left sidebar displays the 'SCHEMAS' tree for a 'my_guitar_shop' database, with tables like 'addresses', 'administrators', 'categories', 'customers', 'order_items', 'orders', 'products', and 'sys'. The main editor window contains a SQL query:

```

1 SELECT
2   IF(GROUPING(c.category_name) = 1,
3     'Grand Total',
4     c.category_name) AS category_name,
5   IF(GROUPING(p.product_name) = 1,
6     'Category Total',
7     p.product_name) AS product_name,
8   SUM(o.product_id) AS total_quantity
9 FROM
10  categories c,
11  products p,
12  order_items o
13 WHERE
14  c.category_id = p.category_id
15  AND p.product_id = o.product_id
16 GROUP BY category_name, product_name WITH ROLLUP
17 HAVING GROUPING(c.category_name) = 1
18      OR GROUPING(p.product_name) = 1;

```

The 'Object Info' panel on the left shows the structure of the 'order_items' table:

Columns:	Item_id	order_id	product_id	item_price	discount_amount	quantity
	int AI PK	int	int	decimal(10,2)	decimal(10,2)	int

The 'Result Grid' shows the query results:

category_name	product_name	total_quantity
Basses	Category Total	7
Drums	Category Total	19
Guitars	Category Total	29
Grand Total	Category Total	55

The 'Action Output' panel at the bottom shows the execution log:

Time	Action	Response	Duration / Fetch Time
37 16:40:21	SELECT IF(GROUPING(c.category_name) = 1, 'Category',...	Error Code: 1054. Unknown column 'o.product_id' in 'field list'	0.00042 sec
38 16:42:18	SELECT IF(GROUPING(c.category_name) = 1, 'Category',...	Error Code: 1054. Unknown column 'o.product_id' in 'field list'	0.00070 sec
39 16:43:40	SELECT IF(GROUPING(c.category_name) = 1, 'Category',...	4 row(s) returned	0.0021 sec / 0.00001...
40 16:46:46	SELECT IF(GROUPING(c.category_name) = 1, 'Grand Total',...	Error Code: 1054. Unknown column 'p.product_name' in 'having clause'	0.00038 sec
41 16:47:02	SELECT IF(GROUPING(c.category_name) = 1, 'Grand Total',...	4 row(s) returned	0.00089 sec / 0.000...

The status bar at the bottom indicates 'Query Completed'.

Figure 7 shows a query which selects the category name, product name and creates a column called `total_quantity` which is the aggregate sum of the products from each category. Essentially this query returns the total quantity of products which were purchased from each category. This also includes a grand total row which is created by the `WITH ROLLUP` operator to create a summary of the data and replaces the null values that would be in that row with literals.

Figure 8

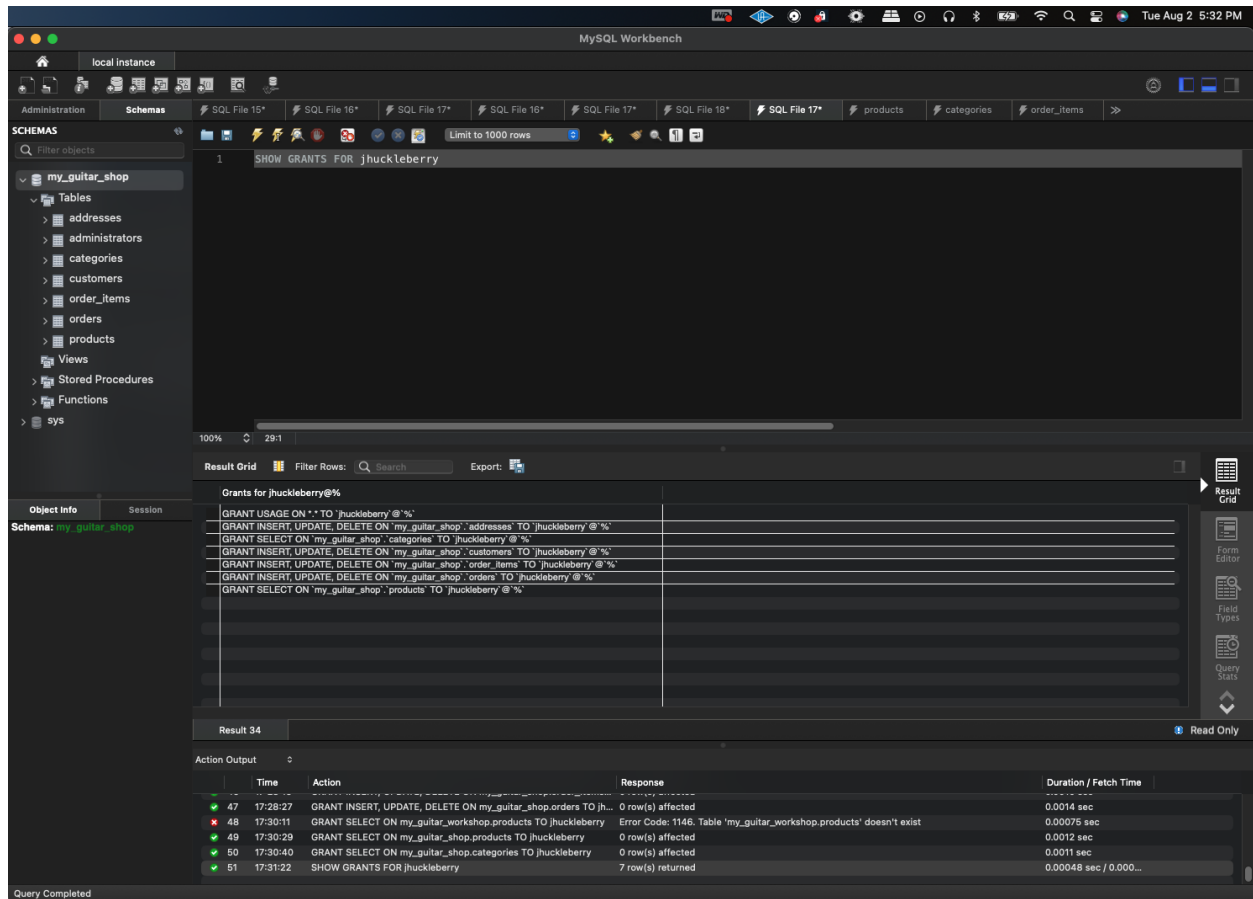


Figure 8 shows a query which shows the granted privileges to a user which was created with my first initial and last name. This user has a password and can access MySQL from any computer. The user has insert, update and delete privileges for the addresses, customers, orders and order items tables of the my guitar shop database. This user also has privileges to use the select operator on the categories and products tables within the my guitar shop database.