

Lab 4: How to Retrieve Data From Two or more Tables

Step 1: Write a SELECT statement that joins the Categories table to the Products table and returns these columns: category_name, product_name, list_price. Sort the result set by the category_name column and then by the product_name column in ascending sequence.

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

```
1 SELECT
2     c.category_name,
3     p.product_name,
4     p.list_price
5 FROM
6     categories c
7 INNER JOIN
8     products p
9 ON
10    c.category_id = p.category_id
11 ORDER BY
12     c.category_name ASC,
13     p.product_name ASC;
```

The Results tab shows the following data:

| category_name | product_name | list_price |
|---------------|--------------------------------------|------------|
| Basses | Fender Precision | 799.99 |
| Basses | Hofner Icon | 499.99 |
| Drums | Ludwig 5-piece Drum Set with Cymbals | 699.99 |
| Drums | Tama 5-Piece Drum Set with Cymbals | 799.99 |
| Guitars | Fender Stratocaster | 699.00 |
| Guitars | Gibson Les Paul | 1199.00 |
| Guitars | Gibson SG | 2517.00 |
| Guitars | Rodriguez Caballero 11 | 415.00 |
| Guitars | Washburn D10S | 299.00 |

The interface also shows a sidebar with navigation options like Server Status, Client Connections, and a bottom status bar indicating the query is completed.

Step 2: Write a SELECT statement that joins the Customers table to the Addresses table and returns these columns: first_name, last_name, line1, city, state, zip_code. Return one row for each address for the customer with an email address of allan.sherwood@yahoo.com.

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

```
1 SELECT
2   c.first_name,
3   c.last_name,
4   a.line1,
5   a.city,
6   a.state,
7   a.zip_code
8 FROM
9   customers c
10  INNER JOIN
11  addresses a
12  ON
13  c.customer_id = a.customer_id
14  WHERE
15  c.email_address = 'allan.sherwood@yahoo.com';
```

Below the query editor, the 'Result Grid' displays the results of the query. It shows two rows of data:

| first_name | last_name | line1 | city | state | zip_code |
|------------|-----------|-------------------------|----------------|-------|----------|
| Allan | Sherwood | 100 East Ridgewood Ave. | Paramus | NJ | 07652 |
| Allan | Sherwood | 21 Rosewood Rd. | Woodcliff Lake | NJ | 07677 |

The interface also includes a left-hand sidebar with navigation options like 'MANAGEMENT', 'INSTANCE', and 'PERFORMANCE'. A right-hand pane shows 'SQLAdditions' with a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

Step 3: Write a SELECT statement that joins the Customers table to the Addresses table and returns these columns: first_name, last_name, line1, city, state, zip_code. Return one row for each customer, but only return addresses that are the shipping address for a customer.

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

```
1 SELECT
2   c.first_name,
3   c.last_name,
4   a.line1,
5   a.city,
6   a.state,
7   a.zip_code
8 FROM
9   customers c
10  INNER JOIN
11  addresses a
12  ON
13  c.shipping_address_id = a.address_id;
```

The Results window displays the following data:

| first_name | last_name | line1 | city | state | zip_code |
|------------|-----------|-------------------------|---------------|-------|----------|
| Allan | Sherwood | 100 East Ridgewood Ave. | Paramus | NJ | 07652 |
| Barry | Zimmer | 16285 Wendell St. | Omaha | NE | 68135 |
| Christine | Brown | 19270 NW Cornell Rd. | Beaverton | OR | 97006 |
| David | Goldstein | 186 Vermont St. | San Francisco | CA | 94110 |
| Erin | Valentino | 6982 Palm Ave. | Fresno | CA | 93711 |
| Frank Lee | Wilson | 23 Mountain View St. | Denver | CO | 80208 |

The Output window shows the following message:

```
# Time Action Message Duration / Fetch
43 11:19:52 SELECT c.first_name, c.last_name, a.line1, a.city, a.state, a.zip_code FROM ... 2 row(s) returned 0.031 sec / 0.000 sec
```

Step 4: Write a SELECT statement that joins the Customers, Orders, Order_Items, and Products tables. This statement should return these columns: last_name, first_name, order_date, product_name, item_price, discount_amount, and quantity. Use aliases for the tables. Sort the final result set by the last_name, order_date, and product_name columns.

The screenshot shows the MySQL Workbench interface. The left sidebar contains the 'MANAGEMENT' section with options like Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, and Data Import/Restore. Below this is the 'INSTANCE' section with Startup / Shutdown, Server Logs, and Options File. The 'PERFORMANCE' section includes Dashboard, Performance Reports, and Performance Schema Setup. The 'Administration' section has Schemas. The main window displays a SQL query in the 'SQL File 3' editor. The query is a SELECT statement that joins the Customers, Orders, and Order_Items tables. The result set is displayed in the 'Result Grid' at the bottom, showing columns: last_name, first_name, order_date, product_name, item_price, discount_amount, and quantity. The results are sorted by last_name, order_date, and product_name. The bottom status bar shows 'Query Completed'.

```
1 SELECT
2   c.last_name,
3   c.first_name,
4   o.order_date,
5   p.product_name,
6   oi.item_price,
7   oi.discount_amount,
8   oi.quantity
9 FROM
10  customers AS c
11 INNER JOIN
12  orders AS o
13 ON
14  c.customer_id = o.customer_id
15 INNER JOIN
```

| last_name | first_name | order_date | product_name | item_price | discount_amount | quantity |
|-----------|------------|---------------------|--------------------------------------|------------|-----------------|----------|
| Zimmer | Barry | 2015-03-28 11:23:20 | Yamaha PG700S | 489.99 | 186.20 | 1 |
| Goldstein | David | 2015-03-31 05:43:11 | Washburn D10S | 299.00 | 0.00 | 1 |
| Valentino | Erin | 2015-03-31 18:37:22 | Washburn D10S | 299.00 | 0.00 | 1 |
| Hernandez | Gary | 2015-04-02 11:26:38 | Tama 5-Piece Drum Set with Cymbals | 799.99 | 120.00 | 1 |
| Sherwood | Allan | 2015-03-29 09:44:58 | Rodriguez Caballero 11 | 415.00 | 161.85 | 1 |
| Wilson | Frank Lee | 2015-04-01 23:11:12 | Ludwig 5-piece Drum Set with Cymbals | 699.99 | 210.00 | 1 |

Step 5: Write a SELECT statement that returns the product_name and list_price columns from the Products table. Return one row for each product that has the same list price as another product. Hint: Use a self-join to check that the product_id columns aren't equal but the list_price columns are equal. Sort the result set by the product_name column.

The screenshot shows the MySQL Workbench interface. The central pane displays a SQL query for a self-join on the 'products' table. The query is as follows:

```
1 SELECT
2   p1.product_name,
3   p1.list_price
4 FROM
5   products AS p1
6 INNER JOIN
7   products AS p2
8 ON
9   p1.product_id < p2.product_id
10  AND p1.list_price = p2.list_price
11 ORDER BY
12   p1.product_name;
```

Below the query editor, the 'Result Grid' is visible, showing the columns 'product_name' and 'list_price'. The first row of data is 'Fender Precision' with a list price of '799.99'. To the right of the query editor, a 'SQLAdditions' pane contains a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

At the bottom, the 'Output' pane shows the execution log:

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--------|---|-----------------------|
| 50 | 11:52:10 | SELECT | p1.product_name, p1.list_price FROM products AS p1 INNER JOIN prod... 2 row(s) returned | 0.032 sec / 0.000 sec |
| 51 | 11:52:42 | SELECT | p1.product_name, p1.list_price FROM products AS p1 INNER JOIN prod... 1 row(s) returned | 0.031 sec / 0.000 sec |

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

Step 6: Write a SELECT statement that returns these two columns: category_name. The category_name column from the Categories table product_id The product_id column from the Products table Return one row for each category that has never been used. Hint: Use an outer join and only return rows where the product_id column contains a null value.

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

```
1 SELECT
2   c.category_name,
3   p.product_id
4 FROM
5   categories AS c
6 LEFT JOIN
7   products AS p
8 ON
9   c.category_id = p.category_id
10 WHERE
11   p.product_id IS NULL;
```

The Results window shows the following columns: category_name, product_id. The data is empty, indicating no rows were returned.

The Output window shows the execution log:

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--------|---|-----------------------|
| 52 | 11:56:15 | SELECT | p1.product_name, p1.list_price FROM products AS p1 INNER JOIN prod... 1 row(s) returned | 0.031 sec / 0.000 sec |
| 53 | 11:58:38 | SELECT | c.category_name, p.product_id FROM categories AS c LEFT JOIN produ... 1 row(s) returned | 0.015 sec / 0.000 sec |

The bottom status bar indicates "Query Completed".

Step 7: Use the UNION operator to generate a result set consisting of three columns from the Orders table: ship_status A calculated column that contains a value of SHIPPED or NOT SHIPPED order_id The order_id column order_date. The order_date column If the order has a value in the ship_date column, the ship_status column should contain a value of SHIPPED. Otherwise, it should contain a value of NOT SHIPPED. Sort the final result set by the order_date column.

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

```
1 -- Orders with ship_date (SHIPPED)
2 SELECT
3     'SHIPPED' AS ship_status,
4     order_id,
5     order_date
6 FROM
7     orders
8 WHERE
9     ship_date IS NOT NULL
10
11 UNION
12
13 -- Orders without ship_date (NOT SHIPPED)
14 SELECT
15     'NOT SHIPPED' AS ship_status,
16     order_id,
```

The result grid displays the following data:

| ship_status | order_id | order_date |
|-------------|----------|---------------------|
| SHIPPED | 4 | 2015-03-30 15:22:31 |
| SHIPPED | 5 | 2015-03-31 05:43:11 |
| NOT SHIPPED | 6 | 2015-03-31 18:37:22 |
| SHIPPED | 7 | 2015-04-01 23:11:42 |
| NOT SHIPPED | 8 | 2015-04-02 11:26:38 |

The interface also shows a sidebar with navigation options (MANAGEMENT, INSTANCE, PERFORMANCE, Administration, Schemas) and a status bar at the bottom indicating "Query Completed".