- Get That Job!
- Become The Best Candidate
- Increase Your Earnings

TOP 30 SQL INTERVIEW CODING TASKS

WITH WINNING SOLUTIONS

Matthew Urban

TOP 30 SQL INTERVIEW CODING TASKS

WITH WINNING SOLUTIONS

TOP 30 SQL Interview Coding Tasks with Winning Solutions

by Matthew Urban

Published by net-boss net-boss.org

August 2018: First Edition

Proofreader: Alex Solsbery

Illustrator: Matthew Urban

Cover designer: Karolina Kaiser

Composition: Karolina Kaiser (mobisfera.pl)

ISBN 978-83-65477-14-9

Preface

SQL is one of the most common programming languages used to manipulate data. If you are a developer, there is a 99% probability that during a job interview you will be asked to solve a couple of coding tasks from SQL. If you are a recruiter, this book gives you a ready to use set of coding tasks with correct answers.

Matthew Urban
IT specialist, Senior Java developer

Data-set

From the perspective of a recruiter, it is easier to use the same data-set during each job interview as is the case with this book. For simplicity, the database schema of a small e-commerce system is used. It contains the following tables: `customers`, `products`, `suppliers`, `orders` and `order_items`. Each coding task described in this book is based on the data-set given below.

Table 1.1 – Example data-set of `customers` table.

customers

ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE	COUNTRY
1	Boris	Spassky	boris@spassky.com	999-888-123	Russia
2	Akiba	Rubinstein	rubi@chess.com	*NULL*	Poland
3	Bobby	Fischer	bobby@fishcher.com	210-6221-9101-22	USA
4	Jose	Capabalnca	play@capablanca.com	032-345-567-678	Cuba

Table 1.2 – Example data-set of `products` table.

products

ID	NAME	AVAILABLE	PRICE	SUPPLIER
1	Baby diaper	true	12	Brandon
2	Crossland bike	true	780	X-Bikes
3	Bicycle helmet	true	50	X-Bikes
4	Energy drink	false	5	Nutrition-V
5	LED bulb	true	30	Electronics Master

Table 1.3 – Example data-set of `suppliers` table.

suppliers

ID	COMPANY_NAME	CONTACT_NAME	ADDRESS	COUNTRY	EMAIL
1	X-Bikes	Malcom Xavery	New York, Yellow 12	USA	malcolm@xbikes.com
2	Brandon	Veronica Brandon	Chicago, Barbecue 780/2	USA	veronica@brandon.com

Table 1.4 – Example data-set of `orders` table.

orders

ID	ORDER_DATE	SHIPPED_DATE	CUSTOMER_ID
1	2014-11-25	2014-12-02	3
2	2016-12-02	2016-12-09	3
3	2017-02-10	2017-02-18	1
4	2018-03-10	2018-03-19	4
5	2019-09-20	2019-09-30	4

Table 1.5 – Example data-set of `order_items` table.

order_items

ORDER_ID	PRODUCT_ID	QUANTITY	PRICE	DISCOUNT
1	2	1	780	0
1	3	1	50	0
2	3	2	50	50
3	5	1	30	30

1. Get data from the database.

The easiest way to verify if a person knows the basics of SQL is to ask them to retrieve data from a database.

Solution

Given three tables: `customers`, `products` and `orders` you need to create queries which retrieve all rows from them. Listing 1.1 presents simple `SELECT` statements.

Listing 1.1 – Example of simple `SELECT` statements.

```
SELECT * FROM customers;

SELECT * FROM products;

SELECT * FROM orders;
```

Please notice that the wildcard (`*`) causes all columns to be retrieved. In many cases, it is necessary to retrieve only part of the data. Listing 1.2 presents an example `SELECT` statement which retrieves only the first and last name of a customer.

Listing 1.2 – The `SELECT` statement which gets specified columns.

```
SELECT first_name, last_name FROM customers;
```

2. Get data from the database using a conditional statement.

Preparing conditional statements is one of the necessary skills every programmer must have. A developer needs to create queries which return only those records that fulfill a specified condition. To retrieve filtered data, the `WHERE` clause combined with `AND`, `OR` and `NOT` operators should be used.

WHERE

First, you are asked to prepare a query which returns all customers which are from the USA.

Listing 2.1 – Example of `SELECT` statement with `WHERE` clause.

```
SELECT * FROM customers
  WHERE country = 'USA';
```

OR

Second, you are asked to prepare a query which returns all customers which are from the USA or Canada.

Listing 2.2 – Example of `SELECT` statement with `OR` operator.

```
SELECT * FROM customers
  WHERE country = 'USA' OR country = 'Canada';
```

AND

Finally, the last most basic operator. You need to prepare a query which returns all products from supplier 'Brandon' and price lower than $20.

Listing 2.3 – Example of `SELECT` statement with `AND` operator.

```
SELECT * FROM products
  WHERE supplier = 'Brandon' AND price < 20;
```

3. Get data from the database using the IN operator.

The `IN` operator is very often used in `SELECT` statements. The `IN` operator can be seen as shorthand for multiple `OR` conditions, but it can also take the results from other `SELECT` queries as input.

List of values

You are asked to prepare a query which returns all customers which are from the following list of countries:

- USA,
- Canada,
- Australia,
- Great Britain,
- New Zealand.

Listing 3.1 presents the correct implementation of such a query.

Listing 3.1 – Example of `SELECT` statement with `IN` operator.

```
SELECT * FROM customers
  WHERE country IN ('USA', 'Canada', 'Australia' , 'Great Britain',
'New Zealand');
```

Subquery

Another way to use the `IN` clause is to pass a list of values by selecting data from another table. For example, you may be asked to retrieve products which were sold in quantities higher than 100. Listing 3.2 presents an example of such a query. From the `order_items` table, you retrieve a list of products identifiers which sold more than 100 items in one order. Next, such a list is passed to the `IN` clause. Finally, the `SELECT` statement returns all products which match previously selected identifiers.

Listing 3.2 – Example of subquery.

```
SELECT * FROM products
  WHERE id IN (SELECT product_id FROM order_items
                 WHERE quantity > 100);
```

4. Save data in the database.

A list of most basic data operations contains save, update, delete and read. Each developer needs to be aware of how to save data in the database before he can modify, delete or read it. To put new rows into an existing table, the SQL provides the `INSERT INTO` statement.

INSERT INTO

In most cases during a job interview, you are asked to save a new customer and a new product in the database. Listing 4.1 presents a correct solution.

Listing 4.1 – Examples of `INSERT INTO` statement.

```
INSERT INTO customers (first_name, last_name, email, phone, country)
  VALUES ('John', 'Malkovich', 'malkovich@yahoo.com', NULL, 'Mexico');

INSERT INTO products (name, available, price, supplier)
  VALUES ('Bike', true, 465, 'T&D');
```

5. Modify data in the database.

The `UPDATE` statement is used to modify existing records in the database. The syntax of `UPDATE` allows you to modify selected columns in one or more rows at once. If you do not specify the `WHERE` clause, all records are going to be updated.

UPDATE

You are usually asked to prepare three `UPDATE` queries: one which modifies all records, a second which modifies only a single record and a third which modifies a subset of data. For example, you need to write a query which removes all phone numbers of all customers. Listing 5.1 presents a solution.

Listing 5.1 – Example of `UPDATE` statement which modifies all rows.

```
UPDATE customers
  SET phone = NULL;
```

Next, you need to prepare a query which modifies the name of a product with identifier 6, as presented in Listing 5.2.

Listing 5.2 – Example of `UPDATE` statement which modifies a single row.

```
UPDATE products
  SET name = 'Crossland bike'
  WHERE id = 6;
```

Finally, you are asked to prepare a query which modifies the shipping date of all orders placed on 2018-09-15.

Listing 5.3 – Example of `UPDATE` statement which modifies multiple rows.

```
UPDATE orders
  SET shipping_date = 2018-10-01
  WHERE order_date = 2018-09-15;
```