Here are the details for each task:

1. **Easy**
   - Write a query to convert all product names in the `products` table to uppercase.

     ```
     SELECT UPPER(product_name) AS product_name
     FROM products;
     ```

   - Extract the last 4 digits of the phone numbers in the `customers` table.

     ```
     SELECT SUBSTRING(phone, -4) AS last_4_digits
     FROM customers;
     ```

   - Concatenate the `first_name` and `last_name` columns from the `employees` table, separating them with a comma.

     ```
     SELECT CONCAT(first_name, ', ', last_name) AS full_name
     FROM employees;
     ```

2. **Moderate**
   - Create a new column called `trimmed_address` that removes leading and trailing whitespace from the `address` column in the `customers` table.

     ```
     SELECT TRIM(address) AS trimmed_address
     FROM customers;
     ```

   - Write a `CASE` statement that categorizes orders in the `orders` table as 'Pending', 'Shipped', or 'Delivered' based on the `order_status` column.

     ```
     SELECT order_id,
            CASE order_status
                WHEN 'Pending' THEN 'Pending'
                WHEN 'Shipping' THEN 'Shipped'
                WHEN 'Delivered' THEN 'Delivered'
                ELSE 'Unknown'
            END AS order_status_text
     FROM orders;
     ```

- Replace all occurrences of the word 'Discount' with 'Sale' in the `product_description` column of the `products` table.

```
SELECT REPLACE(product_description, 'Discount', 'Sale') AS
product_description
FROM products;
```

3. **Challenging**
   - Pad the `product_code` column in the `products` table with underscores to ensure a minimum length of 10 characters.

```
SELECT RPAD(product_code, 10, '_') AS padded_product_code
FROM products;
```

   - Write a nested `CASE` statement that categorizes customers in the `customers` table as 'Platinum', 'Gold', 'Silver', or 'Bronze' based on their `total_spend` (e.g., if `total_spend` > $10,000, they are 'Platinum'; if `total_spend` > $5,000, they are 'Gold'; if `total_spend` > $2,000, they are 'Silver'; else they are 'Bronze').

```
SELECT customer_name,
       CASE
           WHEN total_spend > 10000 THEN 'Platinum'
           WHEN total_spend > 5000 THEN 'Gold'
           WHEN total_spend > 2000 THEN 'Silver'
           ELSE 'Bronze'
       END AS customer_tier
FROM customers;
```

   - Split the `tags` column in the `products` table into separate rows and select the unique tags.

```
SELECT DISTINCT tag
FROM (
    SELECT product_name,
           UNNEST(STRING_TO_ARRAY(tags, ',')) AS tag
    FROM products
) AS tag_table;
```

4. **Advanced**
   - Create a function that takes a product name as input and returns the first three characters of the name, padded with leading zeros to ensure a length of 5

characters.

```
CREATE FUNCTION pad_product_code(product_name TEXT)
RETURNS TEXT AS $$
BEGIN
    RETURN LPAD(SUBSTRING(product_name, 1, 3), 5, '0');
END;
$$ LANGUAGE plpgsql;

SELECT product_name, pad_product_code(product_name) AS padded_code
FROM products;
```

- Write a query that extracts the initials of customers from the `first_name` and `last_name` columns in the `customers` table, and then sorts the results in alphabetical order by the initials.

```
SELECT CONCAT(
           UPPER(SUBSTRING(first_name, 1, 1)), '.',
           UPPER(SUBSTRING(last_name, 1, 1)), '.'
       ) AS initials
FROM customers
ORDER BY initials;
```

Remember, these are just sample solutions, and the actual implementation may vary depending on the specific database and table structures you're working with. Let me know if you have any other questions!