Laporan Implementasi SQL

1. Subquery
   Query:

```
MySQL [dbtoko]> SELECT title
    -> FROM books
    -> WHERE price > (SELECT AVG(price) FROM books);
+--------------------------------------+
| title                                |
+--------------------------------------+
| Harry Potter and the Sorcerer's Stone |
+--------------------------------------+
1 row in set (0.019 sec)
```

   - Subquery digunakan disini untuk mendapatkan semua judul buku yang harganya lebih tinggi diantara buku-buku lain.

2. Distinct
   Query:

```
MySQL [dbtoko]> SELECT DISTINCT costumer_id
    -> FROM transactions;
+-------------+
| costumer_id |
+-------------+
|           1 |
|           2 |
+-------------+
2 rows in set (0.028 sec)
```

   - Kegunaaan **DISTINCT** untuk mengambil daftar unik ID pelanggan dari table *transactions,* agar terhindar dari duplikasi.

3. Case
   Query:

```
MySQL [dbtoko]> SELECT title,
    ->        CASE
    ->            WHEN price < 30000 THEN 'Murah'
    ->            WHEN price BETWEEN 30000 AND 50000 THEN 'Standar'
    ->            ELSE 'Mahal'
    ->        END AS price_category
    -> FROM books;
+-------------------------------------+----------------+
| title                               | price_category |
+-------------------------------------+----------------+
| Harry Potter and the Sorcerer's Stone | Murah          |
| The Lord of the Rings               | Murah          |
| Foundation                          | Murah          |
| Dune                                | Murah          |
| 1984                                | Murah          |
| The Great Gatsby                    | Murah          |
+-------------------------------------+----------------+
6 rows in set (0.001 sec)
```

- CASE untuk mengkategorikan harga buku menjadi ‘Murah’, ‘Standar’, ‘Mahal’ sesuai rentang harga.

4. Grup By
   Query:

```
MySQL [dbtoko]> SELECT costumer_id, COUNT(*) AS total_orders
    -> FROM transactions
    -> GROUP BY costumer_id;
+-------------+--------------+
| costumer_id | total_orders |
+-------------+--------------+
|           1 |            2 |
|           2 |            3 |
+-------------+--------------+
2 rows in set (0.007 sec)
```

- GROUP BY untuk mengelompokkan transaksi berdasarkan ID pelanggan dan menghitung total pesanan per pelanggan.

5. Having
   Query:

```
MySQL [dbtoko]> SELECT costumer_id, COUNT(*) AS total_orders
    -> FROM transactions
    -> GROUP BY costumer_id
    -> HAVING COUNT(*) > 1;
+-------------+--------------+
| costumer_id | total_orders |
+-------------+--------------+
|           1 |            2 |
|           2 |            3 |
+-------------+--------------+
2 rows in set (0.008 sec)
```

- HAVING untuk memfilter hasil agregasi. Dalam hal ini, hanya pelanggan dengan lebih dari satu pesanan yang ditampilkan,