1. Pemilihan Data:
a. Jelaskan perbedaan antara perintah SQL SELECT dan SELECT DISTINCT.

jawab: - SELECT : mengambil semua value atau data termasuk data itu duplikat,
-SELECT DISTINCT : Mengambil hanya nilai atau data yang unik, dan menghilangkan nilai duplikat.

b. Bagaimana cara melakukan pengurutan data (sorting) berdasarkan kolom "tanggal" secara
descending?
jawab: SELECT * FROM nama_table ORDER BY tanggal DESC;

2. Filtering dan Kondisi:
a. Gambarkan perbedaan antara penggunaan WHERE dan HAVING dalam perintah SQL.
jawab: WHERE biasanya di gunakan pada proses filtering data sebelum GROUP BY di eksekusi ,dan untuk HAVING digunakan untuk menggantikan WHERE dalam proses filter data saat menggunakan GROUP BY, dengan catatan datanya di agregasi sperti menetukan vaue MAX, MIN, COUNT dll.

b. Bagaimana Anda menulis sebuah query untuk menampilkan semua data dari tabel "orders" yang
memiliki nilai total lebih dari 1000?
jawab: SELECT * FROM orders LIMIT 50;

3. Gabungan Tabel (JOIN):
a. Jelaskan perbedaan antara INNER JOIN dan LEFT JOIN.
jawab:

- INNER JOIN: Mengembalikan data/ record yang memiliki nilai yang cocok di kedua tabel.
- LEFT JOIN : Mengembalikan semua record dari tabel kiri, dan record yang cocok dari tabel kanan

b. Bagaimana Anda akan menggabungkan dua tabel, "customers" dan "orders", berdasarkan kolom
"customer_id"?
jawab: SELECT * FROM customers
INNER JOIN orders ON customer.customer_id = orders.customer_id

4. Grup dan Agregasi:
a. Bagaimana cara menghitung rata-rata (average) dari kolom "harga" dalam tabel "produk"?
jawab: SELECT AVG(harga) from produk
b. Tuliskan perintah SQL untuk menemukan jumlah total pesanan yang dilakukan oleh setiap pelanggan dalam tabel "orders".
jawab:
```
SELECT customer_id,  COUNT(order_id) AS total_pesanan
FROM orders
GROUP BY customer_id
```

5. Subquery:
a. Jelaskan apa itu subquery dan berikan contoh penggunaannya.
jawab:
sering disebut query nested, dimana query yang tertanam atau dideklarasikan di dalam query utama untuk melakukan multiple step, contoh

```
SELECT nama_produk,
  (SELECT AVG(harga) FROM produk) AS  rata_rata_harga
FROM   produk;
```

b. Bagaimana Anda akan menggunakan subquery untuk menemukan produk dengan harga tertinggi dari setiap kategori dalam tabel "products"?
jawab:
```
SELECT category, product_name, price
FROM products p1
WHERE price = (
      SELECT MAX(price)
      FROM products p2
      WHERE p1.category = p2.category
  );
```

6. Transaksi dan Kunci (Transaction and Keys):
a. Apa itu transaksi dalam konteks database? Berikan contoh penggunaannya.
jawab: transaksi dalam database seperti sekumpulan action yang dilakukan bersama-sama. Action ini dapat mencakup penambahan, perubahan, atau penghapusan data dari database. contoh dari transaction database:

```
-- membuat table ‘customer’
CREATE TABLE customer (
   customer_id INT PRIMARY KEY,
   customer_name VARCHAR(50),
   email VARCHAR(100),
   phone_number VARCHAR(15)
);
```

```
-- Insert data 'customer' table
INSERT INTO customer (customer_id, customer_name, email, phone_number)
VALUES
    (1, wahyudin', 'wahyudin@example.com', '+62834567890'),
    (2, Susis', 'susis@example.com', '+62834567891');
```

b. Jelaskan perbedaan antara kunci primer (primary key) dan kunci asing (foreign key).
jawab: Kunci primer (Primary keys) berfungsi sebagai pengidentifikasi unik untuk setiap baris dalam tabel database. Kunci asing (Foreign Key) menghubungkan data dalam satu tabel dengan data di tabel lain. Kolom kunci asing dalam sebuah tabel menunjuk ke kolom dengan nilai unik di tabel lain (biasanya kolom kunci primer) untuk membuat cara referensi silang atau relasi antara dua tabel.

7. Optimasi Query:
a. Apa yang dimaksud dengan indeks dalam database? Mengapa indeks penting untuk optimasi query?
jawab: indeks adalah struktur data yang meningkatkan kecepatan operasi pengambilan data pada tabel database. Indeks bekerja seperti indeks buku, membantu dengan cepat menemukan data tertentu. kenapa indeks penting untuk optimasi karena Pengambilan Data Lebih Cepat, Mengurangi I/O Disk, Peningkatan Performa Kueri dan Sorting and Grouping, Menjaga Integritas Data seperti keunikan nilai dalam kolom (mis., kunci utama) mungkin memerlukan pemeriksaan tambahan.

b. Bagaimana Anda akan menulis query yang efisien untuk mengambil data dari tabel dengan jutaan baris?
jawab: Untuk mengoptimalkan kueri SQL untuk tabel dengan jutaan baris, gunakan indeks, hindari penggunaan SELECT *, gunakan subkueri , optimalkan JOIN, gunakan LIMIT dan OFFSET,

8. Keamanan Database:
a. Jelaskan risiko SQL injection dan bagaimana cara menghindarinya.
jawab: jenis kerentanan keamanan yang terjadi ketika penyerang dapat memanipulasi kueri SQL aplikasi dengan menyuntikkan kode SQL berbahaya. Hal ini dapat menyebabkan akses yang tidak sah,seperti manipulasi data, dan pelanggaran keamanan lainnya. cara mencegahnya: menggunakan parameterized query, stored procedure, input validation, Privilege Principle, Firewall Aplikasi Web (WAF), escape user input

b. Apa itu enkripsi dalam konteks database? Mengapa enkripsi penting untuk keamanan data?
jawab: Enkripsi dalam konteks basis data mengacu pada proses mengubah data sensitif ke dalam format yang aman dan tidak dapat dibaca, sehingga menyulitkan orang yang tidak berwenang/ not privillage untuk mengakses atau memahami

informasi/ data yang asli. kenapa penting menggunakan enkripsi memastikan kerahasiaan data kredensial atau sensitif, mengamankan transmisi data dan eksploitasi data dari penyerang

9. Database NoSQL:
a. Apa perbedaan utama antara database SQL dan NoSQL?
jawab:

- SQL : Menggunakan skema terstruktur (tabel) dengan kolom dan baris
- NoSQL : Menggunakan skema dinamis (dokumen, grafik, key-value, atau wide-column) yang memungkinkan fleksibilitas dalam menyimpan data

b. Berikan contoh kapan Anda akan memilih menggunakan database NoSQL daripada SQL.
jawab: Database SQL cocok untuk aplikasi yang mengutamakan integritas data. Jika memiliki aplikasi yang menangani data penting seperti informasi keuangan, Anda harus menggunakan database relasional untuk memastikan bahwa setiap kueri yang Anda buat akan memberi Anda respons yang benar dan Anda tidak akan kehilangan data apa pun secara tidak sengaja. Dalam hal ini, Anda ingin mendapatkan konsistensi maksimum, mungkin dengan mengorbankan tingkat ketersediaan dibandingkan dengan NoSQL. database SQL karena database tersebut cocok untuk data terstruktur.

Jika mengembangkan aplikasi yang menangani permintaan simultan dalam jumlah besar dari beberapa klien dan harus mempertahankan ketersediaan tingkat tinggi, Anda mungkin ingin menggunakan database NoSQL. Aplikasi tersebut mungkin berupa aplikasi chating atau antrean pesanan yang harus merespons klien secara instan tanpa latensi. Dengan menggunakan database NoSQL mengorbankan tingkat konsistensi data demi ketersediaan tinggi, Anda sebaiknya memilih database NoSQL.