Here’s a detailed breakdown of the SQL concepts you've provided, including the `GROUP BY`, `WHERE`, and `HAVING` clauses, along with key points, syntax examples, and queries:

1. GROUP BY Clause

The `GROUP BY` clause is used to group rows that have the same values in specified columns into summary rows, like aggregating data (e.g., summing up salaries).

- **Key Points:**
  - Used with aggregate functions (e.g., `SUM()`, `COUNT()`, `MIN()`, `MAX()`, `AVG()`).
  - Often used with `SELECT` to produce aggregated results for each group.
  - Grouping happens after filtering data with the `WHERE` clause but before ordering the results with the `ORDER BY` clause.
- **Syntax:**

```
SELECT column1, aggregate_function(column2)
FROM table_name
WHERE condition
GROUP BY column1
ORDER BY column1;
```

- **Example:**

```
SELECT dept_id, SUM(salary) AS salary_sum
FROM emp
GROUP BY dept_id
ORDER BY salary_sum;
```

  This will group employees by their `dept_id` and sum their salaries, ordering the results by the `salary_sum`.

2. WHERE Clause

The `WHERE` clause is used to filter rows based on specific conditions. It is applied before grouping or aggregating data.

- **Key Points:**
  - Filters rows before aggregation (`GROUP BY`) or ordering (`ORDER BY`).
  - Cannot be used with aggregate functions directly; use `HAVING` instead for post-aggregation filtering.
- **Syntax:**

```
SELECT column1, column2
FROM table_name
WHERE condition;
```

- **Example:**

  ```
  SELECT *
  FROM emp
  WHERE address = 'noida';
  ```

  This will return all employees whose address is 'noida'.

3. HAVING Clause

The `HAVING` clause is used to filter the results after the `GROUP BY` clause, typically based on the result of aggregate functions.

- **Key Points:**
  - Works like `WHERE` but is used after the `GROUP BY` clause.
  - Filters data based on aggregated values, such as sums or counts.
- **Syntax:**

  ```
  SELECT column1, aggregate_function(column2)
  FROM table_name
  GROUP BY column1
  HAVING aggregate_function(column2) condition;
  ```

- **Example:**

  ```
  SELECT dept_id, COUNT(*) AS Total
  FROM emp
  GROUP BY dept_id
  HAVING Total > 3;
  ```

  This will group employees by their `dept_id`, count the total number of employees per department, and then filter to show only departments where the total count exceeds 3.

4. Order of SQL Clauses:

In a SQL query, the clauses should be used in the following order:

1. **SELECT**: Choose the columns to display.
2. **FROM**: Specify the table to retrieve the data from.
3. **WHERE**: Filter rows before aggregation or sorting.
4. **GROUP BY**: Group the rows based on certain columns.
5. **HAVING**: Filter groups based on aggregate functions.
6. **ORDER BY**: Sort the final result set.

5. Detailed Queries Explanation:

a. Simple GROUP BY:

```
SELECT dept_id, SUM(salary) AS salary_sum
  FROM emp
```

```
  GROUP BY dept_id
  ORDER BY salary_sum;
```

- Groups employees by `dept_id` and sums up their salaries. The results are ordered by `salary_sum`.

b. **GROUP BY with multiple columns:**

```
SELECT dept_id, address, COUNT(*) AS Total
  FROM emp
  GROUP BY dept_id, address;
```

- Groups employees by both `dept_id` and `address`. For each group, the query counts the number of employees.

c. **Aggregating multiple functions in GROUP BY:**

```
SELECT dept_id, COUNT(*) AS Total, SUM(salary) AS sal_sum,
MIN(salary) AS sal_min, MAX(salary) AS sal_max, AVG(salary) AS
sal_avg
  FROM emp
  GROUP BY dept_id;
```

- Groups employees by `dept_id`, and provides several aggregate functions: total employees, sum, minimum, maximum, and average salary.

d. **Using HAVING for post-aggregation filtering:**

```
SELECT dept_id, address, SUM(salary) AS sal_sum
  FROM emp
  GROUP BY dept_id, address
  HAVING sal_sum > 10000
  ORDER BY sal_sum DESC;
```

- First, groups the data by `dept_id` and `address`, then sums the salaries. The `HAVING` clause filters groups where the sum exceeds 10,000. Finally, the results are ordered in descending order by the salary sum.

e. **WHERE + GROUP BY + HAVING combination:**

```
SELECT dept_id, address, SUM(salary) AS sal_sum
  FROM emp
  WHERE address = 'noida'
  GROUP BY dept_id, address
  HAVING sal_sum > 10000
  ORDER BY sal_sum DESC;
```

- Filters rows where the address is `noida` before grouping. Groups the data by `dept_id` and `address`, sums the salaries, then filters results where the sum exceeds 10,000.

6. **Summary:**

- **GROUP BY**: Used for grouping rows and applying aggregate functions.
- **WHERE**: Used to filter data before grouping.

- **HAVING**: Used to filter data after aggregation.

Understanding when and how to use `GROUP BY`, `WHERE`, and `HAVING` is key to writing efficient SQL queries. You should always remember the order of SQL clauses and how each operates to ensure correct data retrieval.

```
In [ ]:
```