SQL – Functions and Operators

1.

SQL Operators - Filter

WHERE Clause :

- Used to specify a condition while fetching the data from a single table or by joining with multiple tables.
- Not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement, etc.,

emp_id	first_name	last_name	salary
101	Steven	Cohen	10000
102	Edwin	Thomas	15000
103	Harry	Potter	20000

e.g.

```
SELECT * FROM employees WHERE
emp_id=101;
```

emp_id	first_name	last_name	salary
101	Steven	Cohen	10000

The example mentioned above extracts all the columns from the table 'employees' whose emp_id=101

2.

SQL Operators – Logical

Operator	IllustrativeExample	Result
AND	(5<2) AND (5>3)	FALSE
OR	(5<2) OR (5>3)	TRUE
NOT	NOT(5<2)	TRUE

Sample Queries:

```
SELECT * FROM employees WHERE first_name = 'Steven' and salary = 15000;

SELECT * FROM employees WHERE first_name = 'Steven' OR salary =15000;

SELECT * FROM employees WHERE first_name = 'Steven' and salary !=10000;
```

3.

SQL Operators – Comparison

Comparison Operators	
Symbol	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<> or !=	Not equal to

Sample Queries:

```
SELECT * FROM employees WHERE first_name = 'Steven'
AND salary <=10000;

SELECT * FROM employees WHERE first_name = 'Steven'
OR salary >=10000;
```

4.

SQL Operators – Special

Special Operators	
BETWEEN	Checks an attribute value within range
LIKE	Checks an attribute value matches a given string pattern
IS NULL	Checks an attribute value is NULL
IN	Checks an attribute value matches any value within a value list
DISTINCT	Limits values to unique values

Sample Queries:

```
SELECT * FROM employees WHERE salary
between 10000 and 20000;

SELECT * FROM employees WHERE first_name
like 'Steven';

SELECT * FROM employees WHERE salary is
null;

SELECT * FROM employees where salary in
(10000,12000,20000);

SELECT DISTINCT(first_name) from
employees;
```

5.

SQL Operators – Aggregations

Aggregation Functions	
Avg()	Returns the average value from specified columns
Count()	Returns number of table rows
Max()	Returns largest value among the records
Min()	Returns smallest value among the records
Sum()	Returns the sum of specified column values

Sample Queries:

```
SELECT avg(salary) FROM employees;

SELECT count(*)  FROM employees;

SELECT min(salary) FROM employees;

SELECT max(salary) FROM employees;

SELECT sum(salary) FROM employees;
```

6.

SQL GROUP BY Clause

- Arrange identical data into groups.

e.g.,

```
SELECT max(salary),dept_id
FROM employees
GROUP BY dept_id
```

emp_id	first_name	last_name	salary	dept_id
103	Harry	Potter	20000	12
102	Edwin	Thomas	15000	11
101	Steven	Cohen	10000	10
100	Erik	John	10000	12

7.

SQL HAVING Clause

- Used with aggregate functions due to its non-performance in the WHERE clause.
- Must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used.

e.g.,

```
SELECT AVG(salary),dept_id
FROM employees
GROUP BY dept_id
HAVING count(dept_id)>=2
```

employee_ id	first_name	last_name	salary	dept_id
103	Harry	Potter	20000	12
102	Edwin	Thomas	15000	11
101	Steven	Cohen	10000	10
100	Erik	John	10000	12

8.

SQL ORDER BY Clause

- Used to sort output of SELECT statement
- Default is to sort in ASC (Ascending)
- Can Sort in Reverse (Descending) Order with "DESC" after the column name

e.g.,

```
SELECT * FROM employees
ORDER BY salary DESC;
```

employee_id	first_name	last_name	salary
101	Steven	Cohen	10000
102	Edwin	Thomas	15000
103	Harry	Potter	20000

employee_id	first_name	last_name	salary
103	Harry	Potter	20000
102	Edwin	Thomas	15000
101	Steven	Cohen	10000

9.

SQL UNION

- Used to combine the result-set of two or more SELECT statements removing duplicates
- Each SELECT statement within the UNION must have the same number of columns
- The selected columns must be of similar data types and must be in the same order in each SELECT statement
- More than two queries can be clubbed using more than one UNION statement

SQL UNION

Product1

CATEGORY_ID	PRODUCT_NAME
1	Nokia
2	Samsung
3	HP
6	Nikon

Product2

CATEGORY_ID	PRODUCT_NAME
1	Samsung
2	LG
3	HP
5	Dell
6	Apple
10	Playstation

e.g.,

```
SELECT product_name FROM product1
UNION
SELECT product_name FROM
product2;
```

PRODUCT_NAME
Nokia
Samsung
HP
Nikon
LG
Dell
Apple

10.

SQL UNION ALL

- Used to combine the results of two SELECT statements including duplicate rows.
- The same rules that apply to the UNION clause will apply to the UNION ALL operator.

SYNTAX:

```
SELECT col1,col2... FROM table1
UNION ALL
SELECT col1,col2... FROM table2;
```

SQL UNION ALL

Product1

CATEGORY_ID	PRODUCT_NAME
1	Nokia
2	Samsung
3	HP
6	Nikon

Product2

CATEGORY_ID	PRODUCT_NAME
1	Samsung
2	LG
3	HP
5	Dell
6	Apple
10	Playstation

e.g.,

```
SELECT product_name FROM product1
UNION ALL
SELECT product_name FROM
product2;
```

PRODUCT_NAME
Nokia
Samsung
HP
Nikon
Samsung
LG
HP
Dell
Apple