SQL

STRUCTURED QUERY LANGUAGE

"EMPLOYEE DATA ANALYSIS"

Prepared by:

DEVAKI N

SQL and its Importance

SQL is the database tool that is used to create and access the database to support software applications.

- **SQL is important for managing big data, allowing users to easily search, sort and analyze large datasets.**
- **Universal language**
- **Open-Source - Simple to use and learn**
- **Manage Millions of Rows of Data**
- **Technology Evolution and Increasing Demand**

TABLES DESCRIPTION

emp_id

first_name

last_name

gender

Job

dept

exp

country

continent

salary

hourly_pay

total_salary

1. Basic SELECT with WHERE and ORDER BY

```
SELECT emp_id, first_name, last_name, job,
salary FROM emp_data
WHERE salary > 9000
ORDER BY salary DESC;
```

emp_id	first_name	last_name	job	salary
1	Arthur	Black	CEO	16500
2	Cynthia	Brooks	PRESIDENT	14500
428	Pete	Allen	MANAGER	11000
103	Emily	Grove	MANAGER	10500
583	Janet	Hale	MANAGER	10000
83	Patrick	Voltz	MANAGER	9500

2. GROUP BY with aggregate function

```
SELECT dept, COUNT(*) AS total_employees,
AVG(salary) AS avg_salary FROM emp_data
GROUP BY dept
ORDER BY avg_salary DESC;
```

dept	total_employees	avg_salary
ALL	2	15500.0000
FINANCE	2	9500.0000
AUTOMOTIVE	3	8100.0000
HEALTHCARE	4	6925.0000
RETAIL	5	6500.0000

3. JOINS

a. INNER JOIN

```
SELECT e1.first_name AS Emp1, e2.first_name AS
Emp2, e1.dept FROM emp_data e1
INNER JOIN emp_data e2 ON e1.dept = e2.dept
AND e1.emp_id <> e2.emp_id
ORDER BY e1.dept;
```

Emp1	Emp2	dept
Cynthia	Arthur	ALL
Arthur	Cynthia	ALL
Pete	William	AUTOMOTIVE
Claire	William	AUTOMOTIVE
Pete	Claire	AUTOMOTIVE
William	Claire	AUTOMOTIVE
Claire	Pete	AUTOMOTIVE
William	Pete	AUTOMOTIVE
Emily	Eric	FINANCE
Eric	Emily	FINANCE

b. LEFT JOIN

```
SELECT e.emp_id, e.first_name, e.salary,
e.hourly_pay FROM emp_data e
LEFT JOIN emp_data e2 ON e.emp_id =
e2.emp_id WHERE
e.hourly_pay=e2.hourly_pay;
```

emp_id	first_name	salary	hourly_pay
260	devi	7000	3
620	Katrina	3000	1
57	Dorothy	7700	1
10	William	9000	2
478	David	4000	4
5	Eric	8500	3
52	Dianna	5500	5
505	Chad	5000	2
532	Claire	4300	1
83	Patrick	9500	5

C. RIGHT JOIN

SELECT COALESCE(e2.emp_id, 0) AS emp_id, e2.first_name FROM emp_data e1 RIGHT JOIN emp_data e2 ON e1.emp_id = e2.emp_id;

emp_id	first_name
260	devi
620	Katrina
57	Dorothy
10	William
478	David
5	Eric
52	Dianna
505	Chad
532	Claire
83	Patrick

4. Subquery: Highest salary in each department

```
SELECT emp_id, first_name, dept, salary
FROM emp_data WHERE (dept, salary) IN (
SELECT dept, MAX(salary)    FROM
emp_data    GROUP BY dept);
```

emp_id	first_name	dept	salary
83	Patrick	HEALTHCARE	9500
583	Janet	RETAIL	10000
103	Emily	FINANCE	10500
428	Pete	AUTOMOTIVE	11000
1	Arthur	ALL	16500

5. Aggregate functions: Total and average salary per continent

SELECT continent, SUM(salary) AS total_salary, AVG(salary) AS avg_salary FROM emp_data GROUP BY continent;

continent	total_salary	avg_salary
ASIA	18500	6166.6667
NORTH AMERICA	77700	9712.5000
EUROPE	24300	8100.0000
SOUTH AMERICA	14000	7000.0000

6. Create a view: High earners

CREATE VIEW high_earnerss AS SELECT emp_id, first_name, last_name, salary, country FROM emp_dataWHERE salary > 9000;

SELECT * FROM high_earnerss;

emp_id	first_name	last_name	salary	country
83	Patrick	Voltz	9500	USA
583	Janet	Hale	10000	COLOMBIA
103	Emily	Grove	10500	CANADA
428	Pete	Allen	11000	GERMANY
2	Cynthia	Brooks	14500	CANADA
1	Arthur	Black	16500	USA

7. Indexing to optimize queries on salary

CREATE INDEX idx_salary ON emp_data(salary);

show indexes from emp_data;

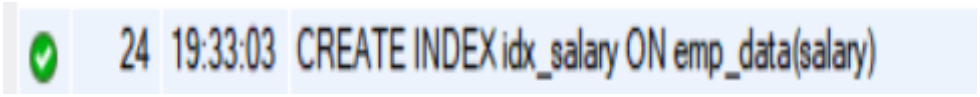

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality
emp_data	1	idx_salary	1	SALARY	A	15

THANK YOU