DBMS Lab Discussion part (Keywords and Practice part)

Lab 2 used Keywords:

SELECT = SELECT is used to select data from a database. It displays the columns.

FROM = FROM command is used to specify which table to select or delete data from.

\* = An asterisk (\*) can be used to specify that the query should return all columns of the queried tables.

DISTINCT = The SELECT DISTINCT statement is used to return only distinct (different) values. Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

DESCRIBE = DESCRIBE command shows the structure of table which include name of the column, data-type of column and the null ability which means, that column can contain null values or not. All of these features of table are described at the time of Creation of table.

AS = The AS command is used to rename a column or table with an alias.

|| = Concatenation operator (||) joins two distinct strings into one string value.

**Practice Part:**

1) The following SELECT statement executes successfully:

SELECT last\_name, job\_id, salary AS Sal FROM employees;

Ans: TRUE.

2) The following SELECT statement executes successfully:

SELECT \* FROM job\_grades;

Ans: False.

3) There are four coding errors in this statement. Can you identify them?

SELECT employee\_id, last\_name sal x 12 ANNUAL SALARY FROM employees;

Ans:

a) The column is named salary, not sal,

b) It should be an asterisk for multiplication instead of x,

c) Since ANNUAL SALARY is two words, it should be in single quotes: 'ANNUAL SALARY',

d) There should be a semicolon after 'employees' in the last line.

4) Show the structure of the DEPARTMENTS table. Select all data from the table.

Ans:

DESCRIBE departments;

SELECT \* FROM departments;

5 & 6) Show the structure of the EMPLOYEES table. Create a query to display the last name, job code, hire date, and employee number for each employee, with employee number appearing first. Save your

SQL statement to a file named lab1\_7.sql. Run your query in the file lab1\_7.sql.

Ans: SELECT employee\_id, last\_name, job\_id,hire\_date AS startdate FROM employees;

7) Create a query to display unique job codes from the EMPLOYEES table.

Ans: SELECT DISTINCT job\_id FROM employees;

8) Display the last name concatenated with the job ID, separated by a comma and space, and name the Column Employee and Title.

Ans: SELECT last\_name ||', '|| job\_id AS "Employee And Title" FROM employees;

9) Create a query to display all the data from the EMPLOYEES table. Separate each column by a comma. Name the column THE\_OUTPUT.

Ans: SELECT employee\_id||','||first\_name||','||last\_name||','||email||','||phone\_number||','||hire\_date||','||job\_id ||','||salary||','||commission\_pct||','||manager\_id||','||department\_id AS "THE\_OUTPUT"FROM employees;

Lab-3 SQL Keyword Discussion:

WHERE = It is used to extract only those records that fulfill a specified condition.

BETWEEN = The SQL BETWEEN operator tests an expression against a range. The range consists of a beginning, followed by an AND keyword and an end expression.

IN = The IN operator checks a value within a set of values separated by commas and retrieve the rows from the table which are matching.

LIKE = The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

a) The percent sign (%) represents zero, one, or multiple characters.

b) The underscore sign (\_) represents one, single character.

NULL = The NULL operator is used to checking if the value of a column is null or not.

AND = Logical AND compares between two Booleans as expression and returns true when both expressions are true.

OR = Logical OR compares between two Booleans as expression and returns true when one of the expression is true.

NOT = Not takes a single Boolean as an argument and changes its value from false to true or from true to false.

ORDER BY = The ORDER BY keyword sorts the records in ascending order by default.

ASC = The ASC command is used to sort the data returned in ascending order.

DESC = To sort the records in descending order, use the DESC keyword.

PRACTISE PART:

1) Create a query to display the last name and salary of employees earning more than $12,000.

Place your SQL statement in a text file named lab2\_1.sql. Run your query.

Ans : SELECT last\_name, salary FROM employees WHERE salary>12000;

2) Create a query to display the employee last name and department number for employee no= 176.

Ans : SELECT last\_name, department\_id FROM employees WHERE employee\_id=176;

3) Modify lab2\_1.sql to display the last name and salary for all employees whose salary is not in

the range of $5,000 and $12,000. Place your SQL statement in a text file named lab2\_3.sql.

Ans : SELECT last\_name, salary FROM employees WHERE salary NOT BETWEEN 5000 AND 12000;

4) Display the employee last name, job ID, and start date of employees hired between February 20,

1998, and May 1, 1998. Order the query in ascending order by start date.

Ans : SELECT last\_name,job\_id, hire\_date FROM employees

WHERE hire\_date BETWEEN '20-FEB-1998' AND '1-MAY-1998'

ORDER by hire\_date;

5) Display the last name and department number of all employees in departments 20 and 50 in alphabetical order by name.

Ans : SELECT last\_name, department\_id FROM employees

WHERE department\_id BETWEEN 20 AND 50

ORDER BY last\_name;

6) Modify lab2\_3.sql to list the last name and salary of employees who earn between $5,000 and

$12,000, and are in department 20 or 50. Label the columns Employee and Monthly Salary, respectively.

Resave lab2\_3.sql as lab2\_6.sql. Run the statement in lab2\_6.sql.

Ans : SELECT last\_name, salary FROM employees WHERE salary BETWEEN 5000 AND 12000 AND

department\_id IN(20,50);

7) Display the last name and hire date of every employee who was hired in 1994.

Ans : SELECT last\_name, hire\_date FROM employees WHERE hire\_date LIKE '94%';

8) Display the last name and job title of all employees who do not have a manager.

Ans : SELECT last\_name, job\_id FROM employees

WHERE manager\_id IS NULL;

9) Display the last name, salary, and commission for all employees who earn commissions. Sort

data in descending order of salary and commissions.

Ans : SELECT last\_name, salary, commission\_pct FROM employees

WHERE commission\_pct IS NOT NULL

ORDER BY salary DESC, commission\_pct DESC;

LAB – 5 KEYWORD DISCUSSION:

SELECT = SELECT is used to select data from a database. It displays the columns.

WHERE = It is used to extract only those records that fulfill a specified condition.

FROM = FROM command is used to specify which table to select or delete data from.

LOWER = The LOWER() function converts a string to lower-case.

UPPER = The UPPER() function converts a string to upper-case.

LENGTH = The LENGTH() function returns the length of a string in a number.

INITCAP = The INITCAP() function is used for setting the first character in word to uppercase and the other to lowercase.

CONCAT = The CONCAT() function adds two or more strings together.

SUBSTR = The SUBSTR() function extracts a substring from a string.

INSTR = The INSTR() function detects the first occurrence of a string or a character in the other string.

LPAD = The LPAD() function used to pad or add a string to the left side of the original string.

RPAD = The RPAD() function used to pad or add a string to the right side of the original string.

ROUND = The ROUND() function rounds a number to a specified number of decimal places.

TRUNC = The TRUNC() function truncates a number to the specified number of decimal places.

MOD = The MOD function is used to get the remainder from a division.

SYSDATE = The SYSDATE function returns the current date and time.

NVL = NVL() converts a null value to an actual value.

MONTHS\_BETWEEN = The MONTHS\_BETWEEN() function is used to get the number of months between dates (date1, date2).

CASE Statement = The CASE statement (WHEN, THEN, CASE, ELSE) goes through conditions and returns a value when the first condition is like an if-then-else statement.

PRACTISE PART:

1) SELECT INITCAP(last\_name), LENGTH(last\_name) FROM employees;

2) SELECT sysdate "Date" FROM dual;

3) SELECT employee\_id, last\_name, salary, ROUND(salary \* 1.15, 0) "New Salary" FROM employees;

4) SELECT employee\_id, last\_name, salary, ROUND(salary \* 1.15, 0) "New Salary", ROUND(salary \* 1.15, 0) - salary "Increase" FROM employees;

5) SELECT INITCAP(last\_name) "Name", LENGTH(last\_name) "Length" FROM employees WHERE last\_name LIKE 'J%' OR last\_name LIKE 'M%' OR last\_name LIKE 'A%' ORDER BY last\_name;

6) SELECT last\_name, ROUND(MONTHS\_BETWEEN(SYSDATE, hire\_date)) MONTHS\_WORKED FROM employees ORDER BY MONTHS\_BETWEEN(SYSDATE, hire\_date);

7) SELECT last\_name, hire\_date, TO\_CHAR(NEXT\_DAY(ADD\_MONTHS(hire\_date, 6),'MONDAY'), 'fmDay, "the" Ddspth "of" Month, YYYY') REVIEW FROM employees;

8) SELECT last\_name, hire\_date, TO\_CHAR(hire\_date, 'DAY') DAY FROM employees ORDER BY TO\_CHAR(hire\_date - 1, 'd');

9) SELECT last\_name, NVL(TO\_CHAR(commission\_pct), 'No Commission') COMM FROM employees;

LAB – 5 KEYWORD DISCUSSION:

SELECT = SELECT is used to select data from a database. It displays the columns.

WHERE = It is used to extract only those records that fulfill a specified condition.

FROM = FROM command is used to specify which table to select or delete data from.

MAX = The MAX() function returns the largest value of the selected column.

MIN = The MIN() function returns the smallest value of the selected column.

AVG = The AVG() function accepts a list of values and returns the average. It can accept both distinct and duplicate values. Only accepts number values.

COUNT = The COUNT() function returns the number of rows that matches a specified criterion.

SUM = The SUM() function returns the total sum of a numeric column.

NVL = NVL() converts a null value to an actual value.

GROUP BY = GROUP BY clause is used with SELECT statement to collect data from multiple records and group the results by one or more columns.

PRACTISE PART:

1) TRUE

2) FALSE

3) TRUE

4) SELECT ROUND(MAX(salary),0) "Maximum salary", ROUND(MIN(salary),0) "Minimum salary", ROUND(SUM(salary),0) "Sum of salaries", ROUND(AVG(salary),0) "Average of salaries" FROM employees;

5) SELECT job\_id, ROUND(MAX(salary),0) "Maximum salary", ROUND(MIN(salary),0) "Minimum salary", ROUND(SUM(salary),0) "Sum of salaries", ROUND(AVG(salary),0) "Average of salaries" FROM employees GROUP BY job\_id;

6) SELECT ROUND(MAX(salary),0) "Maximum", ROUND(MIN(salary),0) "Minimum", ROUND(SUM(salary),0) "Sum", ROUND(AVG(salary),0) "Average" FROM employees;

7) SELECT job\_id, ROUND(MAX(salary),0) "Maximum", ROUND(MIN(salary),0) "Minimum", ROUND(SUM(salary),0) "Sum", ROUND(AVG(salary),0) "Average" FROM employees GROUP BY job\_id;

8) Same as 5 No problem.

9) SELECT job\_id, COUNT(\*) FROM employees GROUP BY job\_id;

10) SELECT d.department\_name "Name", d.location\_id "Location ", COUNT(\*) "Number of People", ROUND(AVG(salary),2) "Salary" FROM employees e, departments d WHERE e.department\_id = d.department\_id GROUP BY d.department\_name, d.location\_id;

11) SELECT COUNT(\*) total, SUM(DECODE(TO\_CHAR(hire\_date, 'YYYY'),1995,1,0))"1995", SUM(DECODE(TO\_CHAR(hire\_date, 'YYYY'),1996,1,0))"1996", SUM(DECODE(TO\_CHAR(hire\_date, 'YYYY'),1997,1,0))"1997", SUM(DECODE(TO\_CHAR(hire\_date, 'YYYY'),1998,1,0))"1998" FROM employees;

Lab-6 Discussion:

CREATE TABLE = The CREATE TABLE statement is used to create a new table in a database.

VARCHAR2 = The VARCHAR2 data type stores alphanumeric values in variable-length strings.

NUMBER = In SQL, numbers are defined as either exact or approximate. The exact numeric data types are SMALLINT, INTEGER, BIGINT, NUMERIC(p,s), and DECIMAL(p,s). Exact SQL numeric data type means that the value is stored as a literal representation of the number's value.

DATE = SQL Server comes with the following data types for storing a date or a date/time value in the database: DATE - format YYYY-MM-DD. DATETIME - format: YYYY-MM-DD HH:MI:SS. SMALLDATETIME - format: YYYY-MM-DD HH:MI:SS. TIMESTAMP - format: a unique number.

DESCRIBE = The DESCRIBE command enables you to describe objects recursively to the depth level set in the SET DESCRIBE command.

ALTER TABLE = The SQL ALTER TABLE command is used to add, delete or modify columns in an existing table.

ADD = The ADD command is used to add a column in an existing table.

MODIFY = The modify command is used when we have to modify a column in the existing table, like add a new one, modify the data type for a column, and drop an existing column.

DROP = DROP is used to delete a whole database or just a table. The DROP statement destroys objects like an existing database, table, index, or view.

INSERT INTO = The INSERT INTO statement is used to insert new records in a table.

VALUES = The VALUES command specifies the values of an INSERT INTO statement.

DELETE = This command is used to delete existing records from a table. Using this, you can either delete specific records based on a condition or all the records from a table.

UPDATE = The UPDATE command in SQL is used to modify or change the existing records in a table. If we want to update a particular value, we use the WHERE clause along with the UPDATE clause. If you do not use the WHERE clause, all the rows will be affected.

TRUNCATE TABLE = TRUNCATE TABLE removes all rows from a table, but the table structure and its columns, constraints, indexes, and so on remain.

PRACTICE PART:

1) CREATE TABLE employee(id NUMBER(4) NOT NULL, last\_name VARCHAR2(25), first\_name VARCHAR(25),userid VARCHAR(8), SALARY NUMBER(9,2));

2) INSERT INTO employee(id, last\_name, first\_name,userid,salary)

VALUES(1,'Patel','Ralph','rpatel',895)

INSERT INTO employee(id, last\_name, first\_name,userid,salary)

VALUES(2,'Dancs','Betty','bdancs',860)

INSERT INTO employee(id, last\_name, first\_name,userid,salary)

VALUES(3,'Biri','Ben','bbiri',1100)

INSERT INTO employee(id, last\_name, first\_name,userid,salary)

VALUES(4,'Newman','Chad','cnewman',750);

3) ALTER TABLE employee MODIFY (last\_name VARCHAR(50)) DESC employee;

4) SELECT \* FROM employee;

5) UPDATE employee SET last\_name='Drexler' WHERE id=3;

6) UPDATE employee SET salary=1000 WHERE salary<900;

7) DELETE FROM employee WHERE last\_name ='Dancs';

8) CREATE TABLE EMP80 AS SELECT \* FROM employees;

9) ALTER TABLE employee DROP COLUMN last\_name;

DESC employee;

10) CREATE TABLE employee2 AS SELECT id employee\_id, first\_name, last\_name, salary FROM employee;

LAB 7 | DISCUSSION:

SELECT = SELECT is used to select data from a database. It displays the columns.

WHERE = It is used to extract only those records that fulfill a specified condition.

FROM = FROM command is used to specify which table to select or delete data from.

DESCRIBE = The DESCRIBE command enables you to describe objects recursively to the depth level set in the SET DESCRIBE command.

ALTER TABLE = The SQL ALTER TABLE command is used to add, delete or modify columns in an existing table.

ADD CONSTRAINT = The ADD CONSTRAINT command is used to create a constraint after a table is already created.

UNIQUE = the UNIQUE constraint ensures that all values in a column are different. A PRIMARY KEY constraint automatically has a UNIQUE constraint.

ADD = The ADD command is used to add a column in an existing table.

OR = The OR command is used with WHERE to include rows where either condition is true.

PRIMARY KEY = The PRIMARY KEY constraint uniquely identifies each record in a table. Primary keys must contain UNIQUE values, and cannot contain NULL values. A table can have only ONE primary key.

FOREIGN KEY = The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables. The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

CHECK = The CHECK constraint limits the value that can be placed in a column.

IN = The IN command allows you to specify multiple values in a WHERE clause.

REFERENCES = The REFERENCES keyword is used to define which table and column are used in a foreign key relationship.

CHECK = The CHECK constraint is used to limit the value range that can be placed in a column.

LIKE = The LIKE command is used in a WHERE clause to search for a specified pattern in a column.

DROP CONSTRAINT = The DROP CONSTRAINT command is used to delete a UNIQUE, PRIMARY KEY, FOREIGN KEY, or CHECK constraint.

PRACTISE PART:

1) ALTER TABLE emp ADD CONSTRAINT my\_emp\_id\_pk PRIMARY KEY (id);

2) ALTER TABLE dept ADD CONSTRAINT my\_dept\_id\_pk PRIMARY KEY(id);

3) ALTER TABLE emp ADD (dept\_id NUMBER(7));

ALTER TABLE emp ADD CONSTRAINT my\_emp\_dept\_id\_fk FOREIGN KEY (dept\_id) REFERENCES dept(id);

4) SELECT constraint\_name, constraint\_type FROM user\_constraints WHERE table\_name IN ('EMP', 'DEPT');

5) SELECT object\_name, object\_type FROM user\_objects WHERE object\_name LIKE 'EMP%' OR object\_name LIKE 'DEPT%';

6) ALTER TABLE EMP ADD commission NUMBER(2,2) CONSTRAINT my\_emp\_comm\_ck CHECK (commission >= 0);

LAB 8 | DISCUSSION:

SELECT = SELECT is used to select data from a database. It displays the columns.

WHERE = It is used to extract only those records that fulfill a specified condition.

FROM = FROM command is used to specify which table to select or delete data from.

JOIN = A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

USING = The USING clause specifies which columns to test for equality when two tables are joined.

AVG() = The AVG() function returns the average value of a numeric column.

NOT NULL = The NOT NULL command is used to test for non-empty values.

ON = ON Clause can be used to join columns that have different names.

RIGHT OUTER JOIN = The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1).

LEFT OUTER JOIN = The LEFT JOIN command returns all rows from the left table, and the matching rows from the right table. It is also known as LEFT OUTER JOIN.

ADD = The ADD command is used to add a column in an existing table.

FULL OUTER JOIN = The FULL OUTER JOIN keyword returns all records when there are a match in left (table1) or right (table2) table records.

PRACTISE PART:

1) SELECT e.last\_name, e.department\_id, d.department\_name FROM employees e, departments d WHERE e.department\_id = d.department\_id;

2) SELECT DISTINCT job\_id, location\_id FROM employees, departments WHERE employees.department\_id = departments.department\_id AND employees.department\_id = 80;

3) SELECT e.last\_name, d.department\_name, d.location\_id, l.city FROM employees e, departments d, locations l WHERE e.department\_id = d.department\_id AND d.location\_id = l.location\_id AND e.commission\_pct IS NOT NULL;

4) SELECT last\_name, department\_name FROM employees, departments WHERE employees.department\_id = departments.department\_id AND last\_name LIKE '%a%';

5) SELECT e.last\_name, e.job\_id, e.department\_id,d.department\_name FROM employees e JOIN departments d ON (e.department\_id = d.department\_id) JOIN locations l ON (d.location\_id = l.location\_id) WHERE LOWER(l.city) = 'toronto';

6) SELECT w.last\_name "Employee", w.employee\_id "EMP#", m.last\_name "Manager", m.employee\_id "Mgr#" FROM employees w join employees m ON (w.manager\_id = m.employee\_id);

7) SELECT w.last\_name "Employee", w.employee\_id "EMP#", m.last\_name "Manager", m.employee\_id "Mgr#" FROM employees w LEFT OUTER JOIN employees m ON (w.manager\_id = m.employee\_id);

LAB 9 | DISCUSSION:

SELECT = SELECT is used to select data from a database. It displays the columns.

WHERE = It is used to extract only those records that fulfill a specified condition.

FROM = FROM command is used to specify which table to select or delete data from.

AND = The AND operator displays a record if all the conditions separated by AND are TRUE.

IN = The IN operator allows you to specify multiple values in a WHERE clause.

ANY = ANY means that the condition will be true if the operation is true for any of the values in the range.

ALL = ALL means that the condition will be true only if the operation is true for all values in the range.

ORDER BY = The ORDER BY keyword is used to sort the result-set in ascending or descending order.

AND = The AND command is used with WHERE only to include rows where both conditions is true.

AVG() = The AVG() function returns the average value of a numeric column.

LEFT JOIN = The LEFT JOIN command returns all rows from the left table, and the matching rows from the right table. It is also known as LEFT OUTER JOIN.

JOIN = The JOIN command returns rows that have matching values in both tables.

NOT NULL = The IS NOT NULL command is used to test for non-empty values.

LOWER() = The LOWER() function converts a string to lower-case.

ON = We use ON clause to specify a join condition. This lets us specify join conditions separate from any search or filter conditions in the WHERE clause.

**PRACTISE PART:**

1) SELECT last\_name, hire\_date FROM employees WHERE department\_id = (SELECT department\_id FROM employees WHERE last\_name = 'Zlotkey') AND last\_name <> 'Zlotkey';

2) SELECT employee\_id, last\_name, salary FROM employees WHERE salary > (SELECT AVG(salary) FROM employees) ORDER BY salary;

3) SELECT employee\_id, last\_name FROM employees WHERE department\_id IN (SELECT department\_id FROM employees WHERE last\_name like '%u%');

4) SELECT last\_name, department\_id, job\_id FROM employees WHERE department\_id IN (SELECT department\_id FROM departments WHERE location\_id = 1700);

5) SELECT last\_name, salary FROM employees WHERE manager\_id = (SELECT employee\_id FROM employees WHERE last\_name = 'King');

6) SELECT department\_id, last\_name, job\_id FROM employees WHERE department\_id IN (SELECT department\_id FROM departments WHERE department\_name = 'Executive');