|  |
| --- |
| Department of Software Engineering  Mehran University of Engineering and Technology, Jamshoro |

|  |  |  |  |
| --- | --- | --- | --- |
| Course: SWE324 - Data Warehousing and Data Mining | | | |
| Instructor | Rabeea Jaffari | **Practical/Lab No.** | 03 |
| Date | 16 April 2019 | **CLOs** | CLO-4: P3 & P4 |
| Signature |  | **Assessment Score** | 1 Mark |

|  |  |
| --- | --- |
| Topic | To familiar with OLTP system query execution |
| Objectives | * To learn query execution in OLTP databases |

|  |
| --- |
| Lab Discussion: Theoretical concepts and Procedural steps |

**OLTP QUERY EXECUTION:**  Since OLTP are operational databases that record daily transactional activities, the typical queries in an OLTP system comprise of retrievals, insertions, updates and deletions. The preferred query language to perform these mentioned tasks is Structured Query Language (SQL).

To get started with SQL, we need to have any SQL Package installed such as MYSQL, SQLSERVER or ORACLE EXPRESS etc.

**SQLSUBLANGUAGES:**

SQL works on the following basic database structures in the databases:

1. Tables
2. Views
3. Indexes
4. Sequences
5. Synonyms

SQL is usually divided into following sub languages to perform operations on the database structures above:

**DDL:** Data definition language handles the definition (which includes creation, modification and deletion) of basic database structures mentioned above). Includes CREATE, ALTER and DROP SQL commands.

**DML:** Data manipulation language handles the data within the database structures and includes SQL commands such as INSERT, UPDATE and DELETE.

**TCL:** Transaction control language involves the commands which manage changes to the database made by the transactions (DML commands). It involves SQL commands such COMMIT for making a transaction permanent or ROLLBACK to undo a DML operation.

**DRL:** Data retrieval language is used to retrieve data from the databases. It involves SELECT SQL command.

**IMPORTANT SQL COMMANDS:** Some of the most important SQL commands include:

1. **SELECT** - extracts data from a database
2. **UPDATE** - updates data in a database
3. **DELETE** - deletes data from a database
4. **INSERT INTO** - inserts new data into a database
5. **CREATE DATABASE** - creates a new database
6. **ALTER DATABASE** - modifies a database
7. **CREATE TABLE** - creates a new table
8. **ALTER TABLE** - modifies a table
9. **DROP TABLE** - deletes a table
10. **CREATE INDEX** - creates an index (search key)
11. **DROP INDEX** - deletes an index

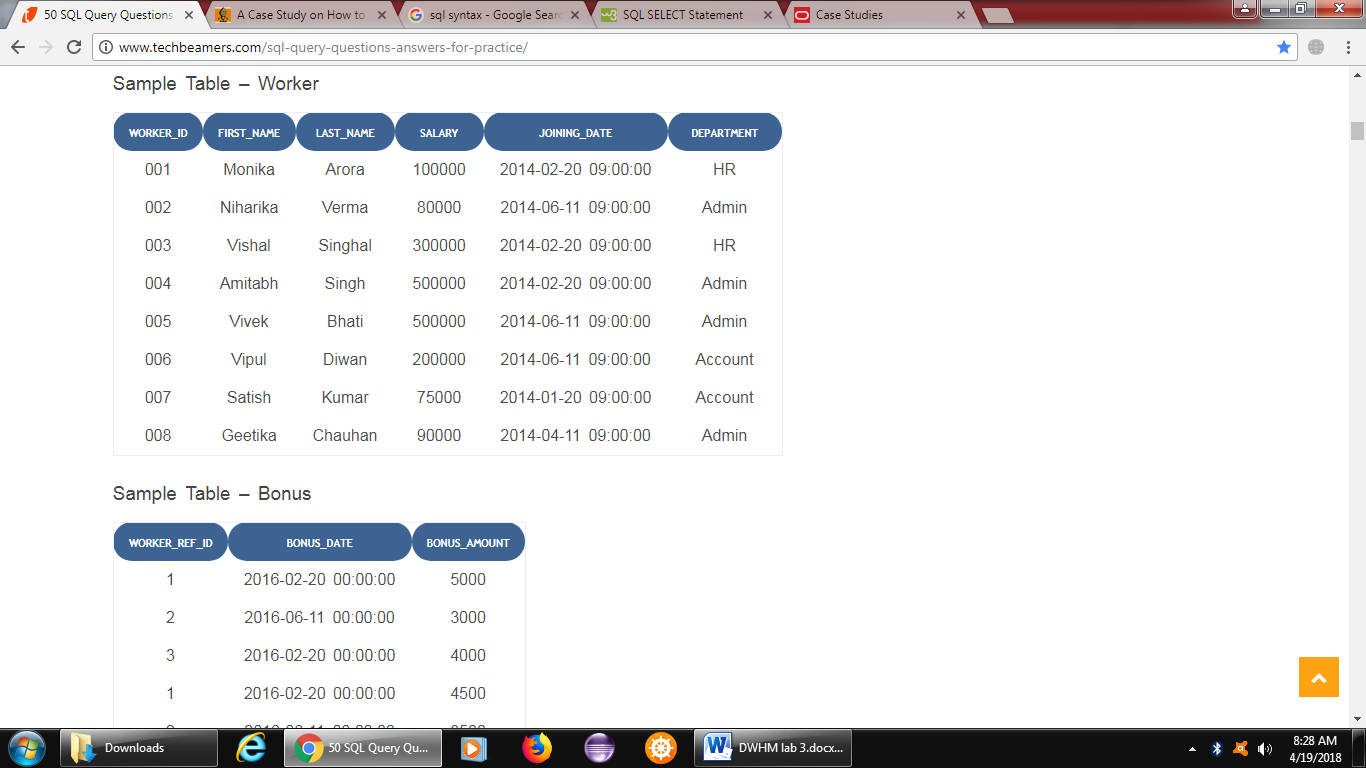
BASIC SQL COMMANDS SYNTAX:

|  |  |
| --- | --- |
| **SQL Statement** | **Syntax** |
| AND / OR | SELECT column\_name(s) FROM table\_name WHERE condition AND|OR condition |
| ALTER TABLE | ALTER TABLE table\_name  ADD column\_name datatype  or  ALTER TABLE table\_name  DROP COLUMN column\_name |
| AS (alias) | SELECT column\_name AS column\_alias FROM table\_name  or  SELECT column\_name FROM table\_name  AS table\_alias |
| BETWEEN | SELECT column\_name(s) FROM table\_name WHERE column\_name BETWEEN value1 AND value2 |
| CREATE DATABASE | CREATE DATABASE database\_name |
| CREATE TABLE | CREATE TABLE table\_name ( column\_name1 data\_type, column\_name2 data\_type, column\_name3 data\_type, ... ) |
| CREATE INDEX | CREATE INDEX index\_name ON table\_name (column\_name)  or  CREATE UNIQUE INDEX index\_name ON table\_name (column\_name) |
| CREATE VIEW | CREATE VIEW view\_name AS SELECT column\_name(s) FROM table\_name WHERE condition |
| DELETE | DELETE FROM table\_name WHERE some\_column=some\_value  or  DELETE FROM table\_name  (**Note:**Deletes the entire table!!)  DELETE \* FROM table\_name  (**Note:**Deletes the entire table!!) |
| DROP DATABASE | DROP DATABASE database\_name |
| DROP INDEX | DROP INDEX table\_name.index\_name (SQL Server) DROP INDEX index\_name ON table\_name (MS Access) DROP INDEX index\_name (DB2/Oracle) ALTER TABLE table\_name DROP INDEX index\_name (MySQL) |
| DROP TABLE | DROP TABLE table\_name |
| EXISTS | IF EXISTS (SELECT \* FROM table\_name WHERE id = ?) BEGIN --do what needs to be done if exists END ELSE BEGIN --do what needs to be done if not END |
| GROUP BY | SELECT column\_name, aggregate\_function(column\_name) FROM table\_name WHERE column\_name operator value GROUP BY column\_name |
| HAVING | SELECT column\_name, aggregate\_function(column\_name) FROM table\_name WHERE column\_name operator value GROUP BY column\_name HAVING aggregate\_function(column\_name) operator value |
| IN | SELECT column\_name(s) FROM table\_name WHERE column\_name IN (value1,value2,..) |
| INSERT INTO | INSERT INTO table\_name VALUES (value1, value2, value3,....)  *or*  INSERT INTO table\_name (column1, column2, column3,...) VALUES (value1, value2, value3,....) |
| INNER JOIN | SELECT column\_name(s) FROM table\_name1 INNER JOIN table\_name2  ON table\_name1.column\_name=table\_name2.column\_name |
| LEFT JOIN | SELECT column\_name(s) FROM table\_name1 LEFT JOIN table\_name2  ON table\_name1.column\_name=table\_name2.column\_name |
| RIGHT JOIN | SELECT column\_name(s) FROM table\_name1 RIGHT JOIN table\_name2  ON table\_name1.column\_name=table\_name2.column\_name |
| FULL JOIN | SELECT column\_name(s) FROM table\_name1 FULL JOIN table\_name2  ON table\_name1.column\_name=table\_name2.column\_name |
| LIKE | SELECT column\_name(s) FROM table\_name WHERE column\_name LIKE pattern |
| ORDER BY | SELECT column\_name(s) FROM table\_name ORDER BY column\_name [ASC|DESC] |
| SELECT | SELECT column\_name(s) FROM table\_name |
| SELECT \* | SELECT \* FROM table\_name |
| SELECT DISTINCT | SELECT DISTINCT column\_name(s) FROM table\_name |
| SELECT INTO | SELECT \* INTO new\_table\_name [IN externaldatabase] FROM old\_table\_name  *or*  SELECT column\_name(s) INTO new\_table\_name [IN externaldatabase] FROM old\_table\_name |
| SELECT TOP | SELECT TOP number|percent column\_name(s) FROM table\_name |
| TRUNCATE TABLE | TRUNCATE TABLE table\_name |
| UNION | SELECT column\_name(s) FROM table\_name1 UNION SELECT column\_name(s) FROM table\_name2 |
| UNION ALL | SELECT column\_name(s) FROM table\_name1 UNION ALL SELECT column\_name(s) FROM table\_name2 |
| UPDATE | UPDATE table\_name SET column1=value, column2=value,... WHERE some\_column=some\_value |
| WHERE | SELECT column\_name(s) FROM table\_name WHERE column\_name operator value |

It should be noted that OLTP systems are not suitable for analytical processing or making decisions therefore business queries are usually difficult to execute using OLTP systems and involve the use of complex joins.

**PRACTICING SQL SKILL:**

Prepare Sample Data To Practice SQL Skill.





**Sample data can be created using the following sql script:**

CREATE DATABASE ORG;

SHOW DATABASES;

USE ORG;

CREATE TABLE Worker (

WORKER\_ID INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

FIRST\_NAME CHAR(25),

LAST\_NAME CHAR(25),

SALARY INT(15),

JOINING\_DATE DATETIME,

DEPARTMENT CHAR(25)

);

INSERT INTO Worker

(WORKER\_ID, FIRST\_NAME, LAST\_NAME, SALARY, JOINING\_DATE, DEPARTMENT) VALUES

(001, 'Monika', 'Arora', 100000, '14-02-20 09.00.00', 'HR'),

(002, 'Niharika', 'Verma', 80000, '14-06-11 09.00.00', 'Admin'),

(003, 'Vishal', 'Singhal', 300000, '14-02-20 09.00.00', 'HR'),

(004, 'Amitabh', 'Singh', 500000, '14-02-20 09.00.00', 'Admin'),

(005, 'Vivek', 'Bhati', 500000, '14-06-11 09.00.00', 'Admin'),

(006, 'Vipul', 'Diwan', 200000, '14-06-11 09.00.00', 'Account'),

(007, 'Satish', 'Kumar', 75000, '14-01-20 09.00.00', 'Account'),

(008, 'Geetika', 'Chauhan', 90000, '14-04-11 09.00.00', 'Admin');

CREATE TABLE Bonus (

WORKER\_REF\_ID INT,

BONUS\_AMOUNT INT(10),

BONUS\_DATE DATETIME,

FOREIGN KEY (WORKER\_REF\_ID)

REFERENCES Worker(WORKER\_ID)

ON DELETE CASCADE

);

INSERT INTO Bonus

(WORKER\_REF\_ID, BONUS\_AMOUNT, BONUS\_DATE) VALUES

(001, 5000, '16-02-20'),

(002, 3000, '16-06-11'),

(003, 4000, '16-02-20'),

(001, 4500, '16-02-20'),

(002, 3500, '16-06-11');

CREATE TABLE Title (

WORKER\_REF\_ID INT,

WORKER\_TITLE CHAR(25),

AFFECTED\_FROM DATETIME,

FOREIGN KEY (WORKER\_REF\_ID)

REFERENCES Worker(WORKER\_ID)

ON DELETE CASCADE

);

INSERT INTO Title

(WORKER\_REF\_ID, WORKER\_TITLE, AFFECTED\_FROM) VALUES

(001, 'Manager', '2016-02-20 00:00:00'),

(002, 'Executive', '2016-06-11 00:00:00'),

(008, 'Executive', '2016-06-11 00:00:00'),

(005, 'Manager', '2016-06-11 00:00:00'),

(004, 'Asst. Manager', '2016-06-11 00:00:00'),

(007, 'Executive', '2016-06-11 00:00:00'),

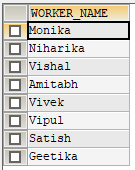
(006, 'Lead', '2016-06-11 00:00:00'),

(003, 'Lead', '2016-06-11 00:00:00');

**SQL QUERIES:**

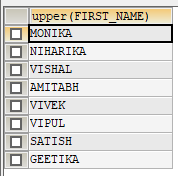
1. Write An SQL Query To Fetch “FIRST\_NAME” From Worker Table Using The Alias Name As <WORKER\_NAME>.

**Solution:** Select FIRST\_NAME AS WORKER\_NAME from Worker;



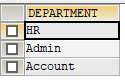
1. Write An SQL Query To Fetch “FIRST\_NAME” From Worker Table In Upper Case.

**Solution:** SELECT UPPER(FIRST\_NAME) FROM worker;



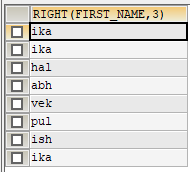
1. Write An SQL Query To Fetch Unique Values Of DEPARTMENT From Worker Table.

**Solution:** SELECT DISTINCT DEPARTMENT FROM worker;



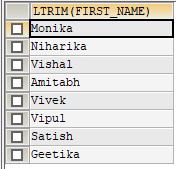
1. Write An SQL Query To Print Last Three Characters Of  FIRST\_NAME From Worker Table.

**Solution:** SELECT RIGHT(FIRST\_NAME,3) FROM worker;



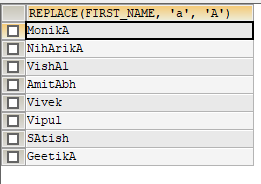
1. Write An SQL Query To Print The FIRST\_NAME From Worker Table After Removing White Spaces From The Left Side.

**Solution:** SELECT LTRIM(FIRST\_NAME) FROM worker;



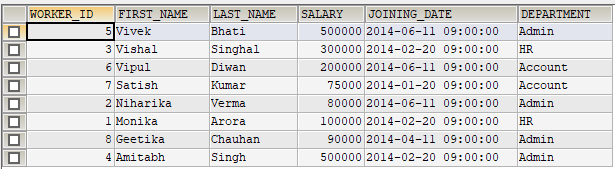
1. Write An SQL Query To Print The FIRST\_NAME From Worker Table After Replacing ‘a’ With ‘A’.

**Solution:** SELECT REPLACE(FIRST\_NAME, 'a', 'A') FROM worker;



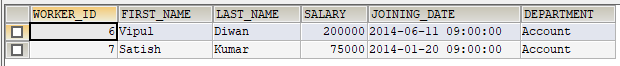
1. Write An SQL Query To Print All Worker Details From The Worker Table Order By FIRST\_NAME Descending.

**Solution:** SELECT \* FROM worker ORDER BY FIRST\_NAME DESC;



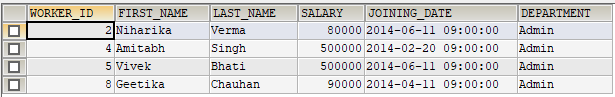
1. Write An SQL Query To Print Details For Workers With The First Name As “Vipul” And “Satish” From Worker Table.

**Solution:** SELECT \* FROM worker WHERE FIRST\_NAME IN('Vipul','Satish');



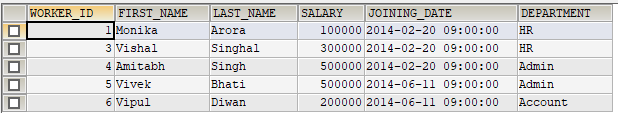
1. Write An SQL Query To Print Details Of Workers With DEPARTMENT Name As “Admin”.

**Solution:** SELECT \* FROM worker WHERE DEPARTMENT = 'Admin';



1. Write An SQL Query To Print Details Of The Workers Whose SALARY Lies Between 100000 And 500000.

**Solution:** SELECT \* FROM worker WHERE SALARY BETWEEN 100000 AND 500000;



1. Write An SQL Query To Print Details Of The Workers Who Have Joined In Jan’2014.

**Solution:** SELECT \* FROM worker WHERE JOINING\_DATE LIKE '2014-01%';



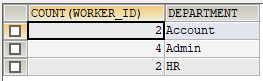
1. Write An SQL Query To Fetch The Count Of Employees Working In The Department ‘Admin’.

**Solution:** SELECT COUNT(WORKER\_ID)FROM worker WHERE DEPARTMENT = "Admin";



1. Write An SQL Query To Fetch The No. Of Workers For Each Department In The Ascending Order.

**Solution:** SELECT COUNT(WORKER\_ID), DEPARTMENT FROM worker GROUP BY DEPARTMENT ASC;



1. Write An SQL Query To Fetch The List Of Employees With The Same Salary.

**Solution:** SELECT w.FIRST\_NAME, w.SALARY FROM worker w INNER JOIN worker v ON v.salary=w.salary WHERE w.worker\_id!=v.worker\_id;



1. Write An SQL Query To Show The Second Highest Salary From A Table.

**Solution:** SELECT MAX(SALARY) FROM worker WHERE SALARY< ( SELECT MAX(SALARY) FROM worker);



|  |
| --- |
| Lab Tasks |
| Submission Date: 26-04-18 |

Execute OLTP queries for the following scenario:

*An organization has a main data table that lists events. Each event has a series of fields including duration, reason code and status.*

*The OLTP system has a table of statuses with attributes status code and Name and there is a common field between the main table (status code), which allows the organization to print the status name rather than the code.*

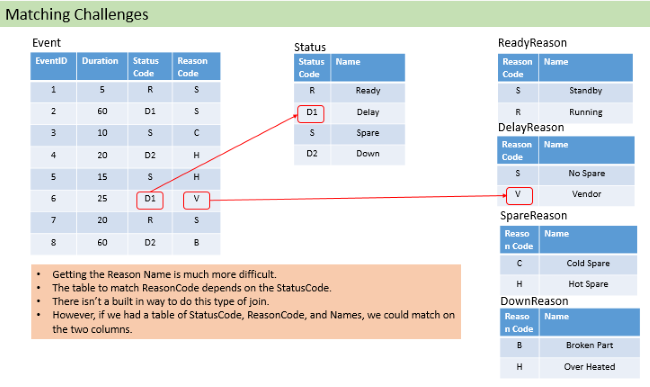
*There are four statuses (ready, delay, spare and down) and there are a range of reasons associated with each status with reason codes. It is possible for the same reason code to exist in two statuses (e.g. delay maintenance and down maintenance).*

*For some reason, the source database has four tables of reasons (one for each status). Each of these has a field which links to the main reason code.*

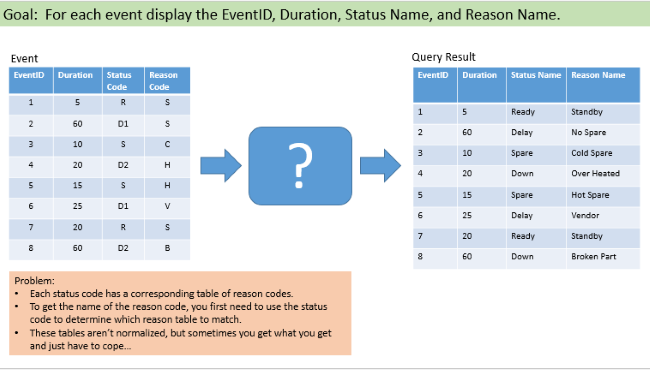
*The organization wishes to do the following:*

*When the main table record status is ‘delay’, then retrieve the name in the ‘delay’ table which corresponds to the main table delay code. When the status is ‘down’, do the same thing but look up the name in the ‘down’ table.*

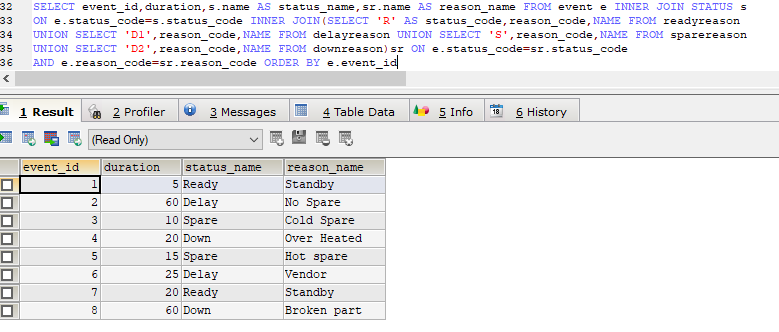
**Sample screenshots:**



Query output should be:



**Hint: Look regarding joins and union in sql.**

**Solution: **