**Final Project Report on SQL**

**Summary of SQL**

Structured Query Language (**SQL**) is a standard computer language for relational database management and data manipulation. **SQL** is used to query, insert, update and modify data stored in a relational database.

SQL statements are used to perform tasks such as **update**data on a **database**, or retrieve data from a **database**

**Database**

A database consists of one or more tables. A table is identified by its name. A table is made up of columns and rows. Columns contain the column name and data type. Rows contain the records or data for the columns.

**Basic SQL**

Each record has a unique identifier or primary key. SQL, which stands for Structured Query Language, is used to communicate with a database.

SQL has five important syntax to create table, select, update, insert, and delete records.

* SELECT - get data from a database table
* UPDATE - change data in a database table
* DELETE - remove data from a database table
* INSERT INTO - insert new data in a database table
* CREATE TABLE – create database table

DELETE, UPDATE, and SELECT statements can use a WHERE clause to limit the rows that are affected or returned and to capture foreign reference relationships.

**SELECT**

The SELECT is used to query the database and retrieve selected data that match the specific criteria that you specify:  
  
SELECT *column1* [, *column2*, ...]   
FROM *tablename*   
WHERE *condition*   
  
The conditional clause can include these operators

* = Equal
* > Greater than
* < Less than
* >= Greater than or equal
* <= Less than or equal
* <> Not equal to
* LIKE pattern matching operator

SELECT \* FROM *tablename*   
  
returns all the data from the table.

Use single quotes around text values (most database systems will also accept double quotes). Numerical values should not be enclosed in quotes.

LIKE matches a pattern. The wildcard *%* is used to denote 0 or more characters.

* 'A%' : matches all strings that start with *A*
* '%a' : matches all strings that end with *a*
* '%a%' : matches all strings that contain an *a*

**CREATE TABLE**

The CREATE TABLE statement is used to create a new table. The format is:   
  
CREATE TABLE *tablename*   
(*column1* *data type*,   
*column2* *data type*,   
*column3* *data type*);

* char(size): Fixed length character string.
* varchar(size): Variable-length character string. Max size is specified in parenthesis.
* number(size): Number value with a max number of columns specified in parenthesis
* date: Date value
* number(size,d): A number with a maximum number of digits of "size" and a maximum number of "d" digits to the right of the decimal

**INSERT VALUES**

Once a table has been created data can be inserted using INSERT INTO command.  
  
INSERT INTO *tablename*   
(col1, ... , coln)   
VALUES (val1, ... , valn)

**UPDATE**

To change the data values in a pre existing table, the UPDATE command can be used.   
  
UPDATE *tablename*   
SET *colX* = *valX* [, *colY* = *valY*, ...]   
WHERE *condition*

**DELETE**

The DELETE command can be used to remove a record(s) from a table.   
  
DELETE FROM *tablename*   
WHERE *condition*   
  
To delete all the records from a table without deleting the table do   
  
DELETE \* FROM *tablename*

**DROP**

To remove an entire table from the database use the DROP command.  
  
DROP TABLE *tablename*

**ORDER BY**

ORDER BY clause can order column name in either ascending (ASC) or descending (DESC) order.  
  
ORDER BY *col\_name* ASC

**AND / OR**

AND and OR can join two or more conditions in a WHERE clause. AND will return data when all the conditions are true. OR will return data when any one of the conditions is true.

**IN**

IN operator is used when you know the exact value you want to return for at least one of the columns   
  
SELECT \* FROM *table\_name* WHERE *col\_name* IN (*val1*, *val2*, ...)

**BETWEEN / AND**

The BETWEEN ... AND operator selects a range of data between two values. These values can be numbers, text, or dates.   
  
SELECT \* FROM *table\_name* WHERE *col\_name* BETWEEN *val1* AND *val2*

**JOIN**

There are times when we need to collate data from two or more tables. That is called a **join**. Tables in a database are related to each other through their keys. We can associate data in various tables without repeating them. For example we could have a table called *Customers* which could have information about customers like their name, address, phone numbers. We could have another table called *Products* that has information regarding the products like part number, product name, manufacturer, number in stock, unit price. A third table called *Orders* could have information regarding what product was ordered, by whom, the date the order was placed, and quantity. Here are the tables:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customers | | | | |
| **Cust\_ID** | **FirstName** | **LastName** | **Address** | **Phone** |
| 01 | Mickey | Mouse | 123 Gouda St. | 456-7890 |
| 02 | Donald | Duck | 325 Eider Ln. | 786-2365 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Products | | | | |
| **Part\_No** | **Name** | **Manufacturer** | **In\_Stock** | **Price** |
| 20-45 | Hammer | Stanley | 57 | 3.50 |
| 21-68 | ScrewDriver | DeVries | 84 | 2.75 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Orders | | | | |
| **Order\_No** | **Part\_No** | **Cust\_ID** | **Date** | **Quantity** |
| 2005-27 | 21-68 | 02 | 31 Oct 2005 | 2 |
| 2005-34 | 20-45 | 01 | 02 Nov 2005 | 3 |

We can obtain information on who has ordered what:   
  
SELECT Customers.FirstName, Customers.LastName, Products.Name   
FROM Customers, Products, Orders   
WHERE Customers.Cust\_ID = Orders.Cust\_ID AND Products.Part\_No = Orders.Part\_No

We can select data from two tables with INNER JOIN. The INNER JOIN returns all rows from both tables where there is a match. If there are rows in *Customers* that do not have matches in *Orders*, those rows will not be listed.   
SELECT Customers.FirstName, Customers.LastName, Orders.Date   
FROM Customers   
INNER JOIN Orders   
ON Customers.Cust\_ID = Orders.Cust\_ID

The LEFT JOIN returns all the rows from the first table (Customers), even if there are no matches in the second table (Orders). If there are rows in *Customers* that do not have matches in *Orders*, those rows also will be listed.   
SELECT Customers.FirstName, Customers.LastName, Orders.Date   
FROM Customers   
LEFT JOIN Orders   
ON Customers.Cust\_ID = Orders.Cust\_ID

The RIGHT JOIN returns all the rows from the second table (Orders), even if there are no matches in the first table (Customers). If there had been any rows in *Orders* that did not have matches *Customers*, those rows also would have been listed.   
SELECT Customers.FirstName, Customers.LastName, Orders.Date   
FROM Customers   
RIGHT JOIN Orders   
ON Customers.Cust\_ID = Orders.Cust\_ID

**ALTER TABLE**

With ALTER TABLE you can add or delete columns in an existing table. When you add a column you must specify a data type.   
ALTER TABLE table\_name   
ADD col\_name datatype   
  
ALTER TABLE table\_name   
DROP COLUMN col\_name

**UNION**

The UNION command is used to select data from two tables very similar to the JOIN command. But the UNION command can be used only with columns having the same datatype. With UNION only distinct values are selected, i.e. if there are common data in the two tables only one instance of that data is returned.   
  
SELECT Name FROM Customers\_USA   
UNION   
SELECT Name FROM Customers\_Asia   
  
This will select all the customers from USA and Asia but if there is a name that occurs in both the tables it will return only one such name. To get all the names use UNION ALL instead.

**GROUP BY**

The GROUP BY was added to SQL so that aggregate functions could return a result grouped by column values.   
  
SELECT col\_name, function (col\_name) FROM table\_name GROUP BY col\_name

HAVING keyword was introduced because the WHERE keyword could not be used. HAVING states a condition.   
  
SELECT clo\_name, function (col\_name) FROM table\_name   
GROUP BY col\_name   
HAVING function (col\_name) condition value