SQL

Most Important SQL Commands

CREATE DATABASE - creates a new database
ALTER DATABASE - modifies a database
CREATE TABLE - creates a new table
ALTER TABLE - modifies a table
DROP TABLE - deletes a table
CREATE INDEX - creates an index (search key)
DROP INDEX - deletes an index
INSERT INTO - inserts new data into a database
SELECT - extracts data from a database
UPDATE - updates data in a database
DELETE - deletes data from a database

CREATE DATABASE - used to create a new SQL database.

```
CREATE DATABASE databasename;
```

DROP DATABASE - used to drop an existing SQL database

```
DROP DATABASE databasename;
```

CREATE TABLE - used to create a new table in a database

```
CREATE TABLE table_name(
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

DROP TABLE - to drop an existing table in a database

```
DROP TABLE table_name;
```

TRUNCATE TABLE - to delete the data inside a table, but not the table itself

```
TRUNCATE TABLE table_name;
```

ALTER TABLE - used to add, delete, or modify columns in an existing table
To add a column in a table

```
ALTER TABLE table_name ADD column_name datatype;
```

To delete a column in a table

```
ALTER TABLE table_name DROP COLUMN column_name;
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

Constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, DEFAULT, INDEX

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);

INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

```
SELECT column1, column2, ...FROM table_name;

SELECT DISTINCT column1, column2, ...FROM table_name;
```

Examples Set 1:

To create a table:

CREATE TABLE `sampledb`.`custtable` (

 `CustId` INT NOT NULL,

 `CustName` VARCHAR(45) NULL,

 `Phone` INT NULL,

 `City` VARCHAR(45) NULL,

 PRIMARY KEY (`CustId`),

 UNIQUE INDEX `CustId_UNIQUE` (`CustId` ASC) VISIBLE)

COMMENT = 'This is to store customer data';

To insert records a table:

INSERT INTO `sampledb`.`custtable` (`CustId`, `CustName`, `Phone`, `City`) VALUES ('101', 'John', '12345', 'Mumbai');

INSERT INTO `sampledb`.`custtable` (`CustId`, `CustName`, `Phone`, `City`) VALUES ('102', 'Jack', '23456', 'NewYork');

To alter a table for adding a new column:

ALTER TABLE `sampledb`.`custtable`

ADD COLUMN `Country` VARCHAR(45) NULL AFTER `City`;

<u>To update an existing table for inserting data in newly added column</u>:

UPDATE `sampledb`.`custtable` SET `Country` = 'India' WHERE (`CustId` = '101');

UPDATE `sampledb`.`custtable` SET `Country` = 'US' WHERE (`CustId` = '102');

UPDATE `sampledb`.`custtable` SET `City` = 'Delhi' WHERE (`CustId` = '101');

<u>To delete an existing table</u>:

use sampledb;

select * from custtable;

drop table custtable;

<u>To retrieve records from a table (**includes where clause, operators, aggregate fn, scalar fn, operators: like, in, between & is null operators)**</u>:

use jobportal;

select * from candidate;

select firstname, lastname from candidate;

select * from candidate where countrycode="us";

select * from candidate where prevworkexperience>5;

select * from candidate where countrycode="us" and prevworkexperience>5;

select count(*) from candidate;

select count(firstname) from candidate;

select count(lastname) from candidate;

select count(*) from candidate where countrycode="us";

select max(testscore) from candidate;

select ucase(firstname) from candidate;

select mid(lastname, 2,3) from candidate;

select firstname, lastname from candidate where contractrecruitercode is null;

select firstname, lastname from candidate where gender="m" and contractrecruitercode is null;

select firstname, lastname from candidate where city in ('NewYork', 'Paris');

```
select firstname, lastname from candidate where prevworkexperience between 6 and 10;

select firstname from candidate where firstname like "%a";

select firstname from candidate where firstname like "%s%";

select firstname from candidate where firstname like "%a_";

select firstname from candidate where firstname like "_a%";

select firstname from candidate where firstname like "__a%";

select firstname from candidate where firstname like "_a_";
```

Order of Existence:

Select -> From -> Where -> GroupBy -> Having -> OrderBy

Precedence of Operations:

From -> Where -> GroupBy -> Having -> Select -> OrderBy

<u>Example Set 2 (OrderBy, GroupBy, Having & Joins)</u>:

```
select firstname, lastname from candidate where contractrecruitercode is not null order by prevworkexperience desc;

select countrycode, count(countrycode) from candidate group by countrycode;

select countrycode, count(countrycode) as CountryCount from candidate group by countrycode having CountryCount>=2;

select countrycode, count(countrycode) as CountryCount from candidate group by countrycode having CountryCount>=2 order by CountryCount;

select countrycode, count(countrycode) as CountryCount from candidate where gender = "m" group by countrycode having CountryCount>=2 order by CountryCount;

select countrycode, count(countrycode) as CountryCount from candidate where prevworkexperience > 4 group by countrycode having CountryCount>=2 order by CountryCount;
```

<u>Joins:</u>

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them
- Relationship between the two tables is established using common columns
- Selects records that have matching values in both tables that undergo join

Inner Join:

SELECT *column_name(s)* **FROM** *table1* **INNER JOIN** *table2* **ON** *table1.column_name = table2.column_name;*

Left Join:

SELECT *column_name(s)* **FROM** *table1* **LEFT JOIN** *table2* **ON** *table1.column_name = table2.column_name;*

Right Join:

SELECT *column_name(s)* **FROM** *table1* **RIGHT JOIN** *table2* **ON** *table1.column_name = table2.column_name;*

Full Outer Join:

SELECT *column_name(s)* **FROM** *table1* **FULL OUTER JOIN** *table2* **ON** *table1.column_name = table2.column_name* **WHERE** *condition;*

Examples Set 3: (Joins)

select c.firstname, cr.name from candidate c join contractrecruiter cr on c.contractrecruitercode = cr.contractrecruitercode;

select c.firstname, cr.name from candidate c left join contractrecruiter cr on c.contractrecruitercode = cr.contractrecruitercode;

select c.firstname, cr.name from candidate c right join contractrecruiter cr on c.contractrecruitercode = cr.contractrecruitercode;

select c.firstname, cr.name from candidate c join contractrecruiter cr on c.contractrecruitercode = cr.contractrecruitercode where cr.percentagecharge>=8;

select c.firstname, cr.name from candidate c join contractrecruiter cr on c.contractrecruitercode = cr.contractrecruitercode where c.prevworkexperience>5 and cr.percentagecharge>=8;

select c.firstname, cr.name, cty.countryname from candidate c join contractrecruiter cr join country cty on c.contractrecruitercode = cr.contractrecruitercode and c.countrycode = cty.countrycode;

Subqueries:

- It is a query within a query that can return individual values or a list of records
- It can be used anywhere an expression is allowed
- outer query is known as the main query, and the inner query is known as a subquery

Rules:

- Subqueries must be enclosed with parentheses
- Subquery can have only one column in Select clause
- A subquery can be placed in many SQL clauses like WHERE, FROM & HAVING
- Subquery can be used with SELECT, UPDATE, INSERT, DELETE statements along with the operators
- Subqueries are on the right side of the comparison operator
- OrderBy cannot be used in Sub-query section, but the main query may use
- GroupBy can be used in Sub-query, but performs the operation of OrderBy
- Sub-query that return more than one row can only be used with multiple value operator like 'IN'
- Between operator cannot be used with subquery, but can be used inside a subquery
- May use IN, NOT IN, All, ANY, SOME, etc for multiple row subquery
- Also uses EXISTS & NOT EXISTS along with WHERE or HAVING

Examples Set 4: (Subqueries)

select firstname from candidate where contractrecruitercode in (select contractrecruitercode from contractrecruiter);

select firstname from candidate where contractrecruitercode in (select contractrecruitercode from contractrecruiter) order by firstname;

select firstname from candidate where contractrecruitercode = any (select contractrecruitercode from contractrecruiter);

select firstname from candidate where contractrecruitercode in (select contractrecruitercode from contractrecruiter where percentagecharge>=8);

select firstname, lastname from candidate where contractrecruitercode in (select contractrecruitercode from contractrecruiter) and companycode in (select companycode from company);

select firstname, lastname from candidate where companycode in (select companycode from company where countrycode in (select countrycode from country));

Stored Procedures:

SP with NO parameter:

CREATE PROCEDURE `SD1234_SP` ()

BEGIN

select * from candidate;

END

call SD1234_SP;

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `SD1234_SP`()
BEGIN
select * from candidate;
select * from contractrecruiter;
select coutryname from country;
END

call SD1234_SP;
```

<u>SP with one IN parameter</u>:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `SD1234_SP`(IN score int)
BEGIN
select * from candidate where testscore > score;
select countryname from country;
END

call SD1234_SP(50);
```

<u>SP with many IN parameters</u>:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `SD1234_SP`(IN score int, ctry varchar(2))
BEGIN
select * from candidate where testscore > score;
select countrycode, count(countrycode) from candidate where countrycode=ctry;
END

call SD1234_SP(50, 'US');
```

<u>SP with OUT parameter</u>:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `SD1234_SP`(OUT maxscore int)
BEGIN
select max(TestScore) into maxscore from candidate;
```

```
END

CREATE DEFINER=`root`@`localhost` PROCEDURE `SD1234_SP`(OUT maxscore int)

BEGIN

select max(TestScore) into maxscore from candidate;

select countryname from country;

END

call SD1234_SP(@score);

select @score;
```

SP with INOUT parameter:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `SD1234_SP`(INOUT maxscore int)

BEGIN

select max(TestScore) into maxscore from candidate where TestScore < maxscore;

select countryname from country;

END

set @score = 50;

call SD1234_SP(@score);

select @score;
```

Views:

- A view is a database object that has no values.
- Its contents are based on the base table.
- It contains rows and columns similar to the real table.
- The View is a virtual table created by a query
- It is operated similar to the base table but does not contain any data of its own.
- The View and table have one main difference that the views are definitions built on top of other tables (or views).

Advantages of View:

1. Simplify complex query
2. Increases the Re-usability
3. Help in Data Security
4. Enable Backward Compatibility.

Example 1:

CREATE OR REPLACE VIEW `SD1234_view` AS

select firstname, lastname, qualification, prevworkexperience from candidate;;

Queries:

select * from sd1234_view;

select firstname from sd1234_view where prevworkexperience > 5;

Common Table Expression:

- A CTE allows you to define a temporary named result set that available temporarily in the execution scope of a statement such as SELECT, INSERT, UPDATE, DELETE, or MERGE.
- You can refer a CTE only within the execution scope of the statement that immediately follows the WITH clause. After you've run your statement, the CTE result set is not available to other statements

Example:

with cteCandidate(CanId, CanName, ConRecrtName) as (

select c.candidatecode, c.firstname, cr.name from candidate c join contractrecruiter cr

on c.contractrecruitercode = cr.contractrecruitercode)

select CanId, CanName, ConRecrtName from cteCandidate;

Ranking:

- Rank function is a window function that could be used to calculate a rank for each row within a partition of a result set.
- The rows within a partition that have the same values will receive the same rank. The rank of the first row within a partition is one. The RANK() function adds the number of tied rows to the tied rank to calculate the rank of the next row, therefore, the ranks may not be consecutive.

Example:

```
select firstname, rank() over (order by firstname) as RankValue from candidate;
```

```
select firstname, candidatecode, contractrecruitercode, rank() over
(partition by contractrecruitercode order by firstname) as RankValue from candidate;
```

Pivot:

- Pivot is used to transform the data in rows to be shown in columns
- It summarizes or consolidates data from table in a different dimension
- It groups data based on some values & shows in result

Example:

```
select productname,
sum(case when suppliercode='s001' then quantity else 0 end) as AKEnterprises,
sum(case when suppliercode='s002' then quantity else 0 end) as TKP,
sum(case when suppliercode='s003' then quantity else 0 end) as PKS
from stock group by productname;
```

```
use jobportal;
select contractrecruitercode,
sum(case when countrycode='us' then prevworkexperience else 0 end) as USA
from candidate group by contractrecruitercode;
```