Sql (Structured Query Language)

↳ **Database**: Organised collection of Data stored in an electronic format.

↳ **DBMS** : A System Software for creating and managing database. it stores data as file eg: xml

↳ **RDBMS** : it stores data into collection of tables, which is related by common fields betn columns of table. eg: Sql Server, oracle, MySql

↳ **Database Architecture**:
- → File Server : DBMS
- → Client Server : RDBMS

* **Constraints** : A Rule associated with a column that the data entered into that column.

① **unique** : No two records can have same value in a particular column.

② **Not Null** : A column can't be left blank.

③ **Primary Key** : A unique identification of each record or row in a table.

Primary Key = Unique + Not Null
Candidate Key = Unique + Null

④ **Foreign Key** : A column or group of columns in a relational database table that provides a link betn data in two tables. It is a column that references a column (mostly often the primary key) of another table.

⑤ **Check** : It is used to ensure whether the value in columns fulfills the specified condition.

* Create Database:

```
create database Intro_SQL;
use Intro_SQL  --> To use database (Activated)
show tables;  --> To see all tables
select * from information_schema.tables;
```

* Create Table:

①
```
create table dep
(
depid int not null,
depname varchar(25),
depadd varchar(100),
PRIMARY KEY (depid)
);
```

②
```
create table emp
(
empid int,
empname varchar(100),
empadd varchar(255),
depid int, age int check(age>=18),
PRIMARY KEY(empid),
FOREIGN KEY(depid) REFERENCES dep(depid));
```

* Insert values into table:

```
INSERT INTO dep values(10, 'MLEU', 'Nanded');
```

* Data Definition Language (DDL): CADT

① Create Table: Column Name should start with letter
② Alter Table: Adding or column deletion
③ Drop Table: Entirely deleting the table including table name/column name which won't happen in the deletion.

* Alter table for adding/droping a column:

① ALTER table emp ADD gender varchar(1) → Added a new column
② ALTER table emp DROP column gender → Delete the column
③ ALTER table emp ADD check (age>=18);

* Drop table:

① drop table emp → droping the table
Truncate (only deletes data of table)
② drop database random_new → droping the database

* Data Query Language (DQL): Data Analysis

① Select: To fetch required data

eg: ① select count(*) as record_count from emp → to get row count
② select count(id) as record_count from emp

* Change table name:

eg: Alter table employee rename to employee_info;/
rename table employee to employee_info;

* Data Manipulation Language : (DML) UID

① Insert : insert or add a row of data into the table
② Update : Update or change records that match a criteria
③ Delete : Delete records or rows from the table

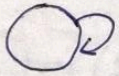
eg : ① insert into emp (first_name, last_name, id, age)
values ('John', 'Mckaethy', 201, 21);

② update emp set designation = 'Sr. Analyst',
age = 45 where emp_name = 'John';

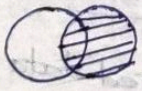
③ delete from emp where first_name = 'John';

* Joins : Joins comes into picture when we might need data from multiple tables.

① Self join : eg. Employee & His Manager.



② Right join / Right outer join :



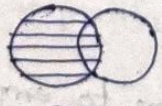
③ Full join / Full outer join :

Full join = left + Right join
use 'union' keyword.

④ Inner join : (simple join)

It selects the rows that satisfies both the table.

⑤ Left join / Left outer join :



⑥ Cross join :

Cartesian product i.e. all possibilities

* Aggregate function :

① groupby() : To aggregate values of column
② count() : count the number of rows
③ min() & max() : Find minimum & maximum values for a particular column
④ avg() : Find the average
⑤ sum(), first(), last()

eg: ① select contract, gender, count(*) as total_row_count from telco_churn group by contract, gender order by contract asc

② select round(sum(totalcharges), 2) from telco_churn

③ select min/max/avg(totalcharges) from telco_churn

* Having clause: WGHOL

↳ Where clause is used before aggregation but having clause works after aggregation becoz where can not be used with aggregate function.

eg: ① select contract, count(*) as cnt from telco_churn group by contract having cnt > 2500 order by contract asc limit 2

* Data Control Language : deal with rights, permissions and other control of DB system
(Grant & Revoke)

eg: Grant/Revoke select, update on employee_info to/from user1;

* String Functions : to make it more understandable for analysis.

① concat : Concatenates two strings
② substr : Extracts a substring from a string
③ upper/lower : convert a string to upper case / lower case
④ character_length : returns length of the string

eg: ① select concat(trim(first_name), ' ', trim(second_name)) from customers

② select substr/(mid)("Sql Tutorial", 3, 8) as extracted_string

* Date & Time Functions :

① Datediff : Returns difference bet^n two dates
② date_format : Returns formatted date variable
③ day/quarter : Returns day or quarter
④ addday : add days to given date

eg: ① select datediff(sysdate(), order_date) from table1
② select date_format(order_date, '%d/%m/%Y') from table1
③ select day/quarter(order_date) from table1
④ select adddate/(subdate)('2017-05-15', interval 10 day/year/quarter);

* Regular Expression :

↳ Regex comes into picture when operator 'like' fails.

↳ Regex are used to search and locate specific sequences of characters that match a pattern.

eg: ① select * from telco_churn where paymentmethod regexp '^e';

② select * from telco_churn where customerID regexp '[vwx]'/'[v-z]'

* Nested Queries :

↳ Nested Queries come into picture when we are dealing with aggregate function becoz aggregate function can't be in 'where' clause.

eg:

① select * from telco_churn where totalcharges > avg(totalcharges) ⟶ Invalid

② select * from telco_churn where totalcharges > (select avg(totalsalaries) from telco_churn)

* Views : Read-only mode

↳ Views are virtual tables that don't store any data of their own but display data stored in other tables.

↳ Advantages :

1. Hide the complexity of data.
2. Act as aggregated tables.
3. Access can be given to views without giving access to tables behind it.
4. Allow massive performance improvements

eg: 1. create view telco_females as
select * from telco_churnna where gender = 'Female';

* Stored Procedures : collection of sql statements that can be used as a function to access DB.

↳ It is a prepared sql code that you can save & reuse it.

eg: create procedure selectingallcustomers as
select * from customers
go;

→ Exceptions are handled using try-catch block.

→ Stored procedure is used to reduce network traffic and improve performance.

* Sql injection :

↳ It is a hacking technique widely used by hackers to steal data from databases or tables.

eg: add malicious code to get username & password from database.

* Triggers : allows to execute a batch of sql code

↳ It is a stored program in DB which automatically gives responses to event of DML operations done by insert, update or delete.

* DCL : Grant or Revoke database access

* Cursor : A control which enables traversal over the rows or records in the table. it was useful for retrieval, addition & removal of DB records.

* Referential Integrity : the consistency must be maintained betn primary & foreign keys i.e. every foreign key value must have a corresponding primary key value.

* Composite Primary Key : It is a set of columns whose values uniquely identify every row in a table.

* Data Integrity : It defines accuracy & consistency of data stored in database.

* Collation: A set of rules that determine how data can be sorted as well as compared.

* Data Warehouse: It refers to a central repository of data where data is assembled from multiple sources of information.
↳ Data Marts: a subset of data warehouse data.

* Entity: can be anything eg: person, place

Relationship: Relationship betn entities

* Transaction Control Language (TCL):

↳ TCL can be used with DML commands like insert, delete and update.

eg: Commit, Rollback, savepoint, set transaction (specifies characteristics for the transaction)

1. savepoint savepoint_name;
2. set transaction [Read write | read only];

↳ Candidate key = unique + Null. A table can have multiple candidate keys. a candidate key which is not a primary key is referred as an 'Alternate key'.

* Connecting to Database :

↳ We can install mysql-connector to connect to databases :

① `! pip install mysql-connector-python`

② `impost mysql.connector as connection`

```
try:
    mydb = connection.connect(host = "localhost",
                              user = "root", paswd = "Root@2580",
                              use-pure = True)
    query = "show databases"/" use employee'
    cursor = mydb.cursor()
    cursor.execute(query)
    print(cursor.fetchall())
except Exception as e:
    mydb.close()
    print(str(e))
```

or

③

```
try:
    df = pd.read_sql('select * from empdetails',
                     con = mydb)
    print(df)
except Exception as e:
    mydb.close()
    print(str(e))
```

* Normalization :

→ It is the process of decomposing a larger, complex table into simple & smaller table in order to remove all redundant data.

① 1NF : Remove all duplicate columns from table

② 2NF : Placing the subsets of data in seperate tables and creation of relationships betn tables using primary keys

③ 3NF : Remove columns which are not dependent on primary key

④ 4NF : No Multivalued dependiencies.

* Index : A performance tuning method of allowing faster retrieval of records from the table.

↳ Index creates an entry for each value and it will be faster to retrieve data.

① Clustered index : It reorders (sorts) the physical order of table and search based on key values. each table can have only one clustered index. eg: Primary Key roll no./id

② Non-clustered index : It doesn't alter the physical order of table and maintains logical order of data. each table can have 999 non-clustered indexes.
eg: index of book i.e. chapter name & page no.

↳ Clustered index is faster, requires less memory for operations.

* Window functions :

↳ A Window is a set of rows in the DB on which an operation is performed.

↳ All the window functions work on this window and hence they are collectively termed as window functions.

↳ they produce an aggregated value for each row. hence each row in a window function maintains its unique identity.

↳ each window function is followed by an 'over clause'.

↳ window functions are faster and scale better than self-joins or cursors.

↳ Types of Window functions:

1. Analytic functions
2. Ranking functions

→ 'Over clause' determines the partitioning and ordering of a set of rows before the associated window function is applied i.e. it defines a window. it accepts following arguments to define a window:

1. partition clause (partition by)
2. order clause (order by)
3. frame clause (range of rows)

① Analytic function :

↳ An analytic function computes values over a group of rows and returns a single result for each row.

↳ this is different from aggregate functions, which return a single value for a group of rows.

↳ it includes over clause, which defines a window of rows around the row being evaluated.

↳ various analytic functions are :
lag(), lead(), first_value(), last_value(), cume_dist(), percent_rank(), percentile_cont(), percentile_disc()

① Lag() → gets value from any desired row that precede the current row.

② lead() → gets value from any desired row that follows the current row.

③ first_value() → returns the first value in an ordered set of values.

④ cume_dist() → used to calculate the cummulative distribution of a value within a group of values. it calculates relative position of a specified value in a group of values.

⑤ percent_rank() → it calculates the percentile ranking of rows in a result set. it ranges from 0 to 1.

② Ranking functions :

↳ it is used to specify rank for individual fields as per the categorization.

eg: table with rows:

[A, B, B, C, D] [E, E, E, F, G]

① Rank = [1, 2, 2, 4, 5] [1, 1, 1, 4, 5]

② dense_rank = [1, 2, 2, 3, 4] [1, 1, 1, 2, 3]

③ row_number = [1, 2, 3, 4, 5] [1, 2, 3, 4, 5]

④ ntile → it distributes the rows in an ordered partition into a specified group of rows.

eg: rank() / dense_rank() / row_number() / ntile() over (partition by x order by y).

↳ Union removes duplicates from query result whereas union all doesn't.

↳ A Null value is a value which is unavailable, unassigned, unknown or not applicable whereas zero is a number, a blank space is treated as a character.

↳ Drop deletes table including table name / column name. Truncate removes all rows & Rollback can't be used to restore whereas delete removes rows aligning with where condition and can be restored using rollback command.

① nth highest salary:

① select name, salary from employee order by salary desc
limit n-1, 1;

② select name, salary from employee as e1
where n-1 = (select count(distinct salary) from
employee as e2 where e2.salary > e1.salary);

③ select * from (select ename, salary,
rank() / dense_rank() / row_number() over(order by/partition by salary desc) r from
employee) where r = n;

④ select * from emp where salary =
(select min(salary) from emp where salary in
(select distinct top n salary from emp order by
salary desc));

↳ row_number() allots ranks in sequential manner,
rank() allocates same rank to same values and
skips a rank number and dense_rank() allocates
same rank to same values without skipping a rank number

⑤ select * from (select dense_rank() over(order by
salary desc) as n, empname, salary from employee)
as max where n = 2;

② Duplicate emails :

① select email from (
select email, count(email) as count from person
group by email) as email_count where count > 1;

② select email from person group by email having
count(email) > 1

③ Find all dates with higher temperature compared to previous day's date:

① select distinct a.id from weather a, weather b
where a.temperature > b.temperature and
datediff (a.recorddate, b.recorddate) = 1;

④ employee with highest salary in each department:

① select dept_id, employee_name, max(salary) from
department group by dept_id;

② select dept_id, emp_name, salary from emp_details
where (dept_id, salary) in (select dept_id,
max(salary) from empdetails group by dept_id);

⑤ query for exchanged seats:

select case
when ((select max(id) from seat) % 2 = 1) and
id = (select max(id) from seat) then id
when id % 2 = 1 then id + 1
else id - 1 end as id, student from seat order by id;

⑥ students who have taken more than one course.

① select distinct e1.student_id from enrolled as e1, enrolled as e2 where e1.student_id = e2.student_id and e1.course_id != e2.course_id;

⑦ display current date:

select GetDate() / sysdate();

⑧ check format of date:

select isdate('1/08/13') as "MM/DD/YY";

⑨ count of employees working in HR department:

select count(*) from emp where dept = 'HR';

⑩ Retrieve first 4 characters of name:

select substring (empname, 1, 4) from emp;

⑪ Fetch only placename before brackets: eg: Hyderabad (hyd)

① select mid (address, 0, locate ('(', address)) from emp;

② select substring (address, 1, charindex ('(', address)) from emp;

(12) create a new table which consists of data and structure copied from other table!

(1) select * into newtable from emp where 1 = 0;

(2) create table newtable as select * from emp;

(13) salary in bet^n 50,000 to 100000:

select * from emp where salary between '50000' and '100000';

(14) employee name begins with 's':

select * from emp where empname like 's%';

(15) fetch top 'N' records:

(1) select top N * from emp order by salary desc;

(2) select * from emp order by salary desc limit N;

(16) create fullname:

select concat (empfname, ' ', emplname) as 'Fullname' from emp;

(17) emp with DOB range and grouped by gender:

select count(*), gender from emp where DOB between '02/05/1970' and '31/12/1975' group by gender;

(18) empname ends with 'a' and contains five alphabets!

```
select * from emp where empname like '____a';
```

(19) Fetch all employees excluding employees with first name sanjay and sonia!

```
select * from emp where empname not in ('sanjay',
'sonia');
```

(20) address like 'Delhi(Del)';

```
select * from emp where address like 'Delhi(Del)%';
```

(21) employee who are managers !

(1) select e.empname, p.empposition from employee e inner join employeeposition p on e.empid = p.empid and p.empposition in ('Manager');

✓(2) select * from emp where (emp_id in (select manager_id from emp));

(3) select e.ename, e.empno, m.ename as manager, e.mgr from emp e, emp m where e.mgr = m.empno;

(4) select * from emp where emp_id in (select manager_id from emp);

(22) dept-wise count of employees:

```
select dept, count(empid) as empdeptcount
from emp group by dept order by empdeptcount asc;
```

(23) Retrieve even or odd records from table:

```
select empid from (select rowno, empid from emp)
where mod(rowno, 2) = 0 → even records
where mod(rowno, 2) = 1 → odd records
```

(24) employee details from emp table who have DOJ in in empposition table:

```
select * from emp e where exists (select *
from empposition p where e.empid = p.empid);
```

(25) Display first & last record in a table:

(1) `select * from emp where empid = (select min(empid)
max(empid) from emp);` → first & last record

(2) `select * from emp where empid <= (select
count(empid)/2 from emp;` → fetch 50% records

(26) employee whose joining year is 2013:

(1) `select * from emp where to_char(joining_date,
'YYYY') = '2013';`

② select * from emp where to_char (joining_date, 'MM') = '01'; or tochar(joining_date, 'MON') = 'JAN';

③ select * from emp where joining_date < to_date ('01/01/2013', 'DD/MM/YYYY'); → employee who joined before january 1st, 2013.

㉗ Delete duplicates!

Delete from emp where emp_id not in (
select max (emp_id) from emp group by empname, dept, contactno, city);