**My SQL**

**create database sql\_dhrumil;**

**use sql\_dhrumil ( DB NAME);**

**SHOW TABLES;**

**DESCRIBE (TABLE NAME);**

**create table employees2 (Emp\_Id int primary key, Emp\_name varchar(25), Age int, Gender char(1), Doj date, Dept varchar(20), city varchar(15), salary float);**

insert into employees2 values

(101, "jimmy", 32, "M", "2005-04-22", "sales", "chicago", 70000),

(102, "sonu", 40, "F", "2006-04-22", "Marketing", "paris", 50000),

(103, "shushma", 32, "F", "2005-04-15", "engg", "mumbai", 45000),

(104, "rahul", 28, "M", "2005-08-27", "Marketing", "Surat", 55000),

(105, "abhi", 26, "M", "2004-04-19", “engg", "bharuch", 66000);

**select \* from employees2;**

**select distinct dept from employees2;**

**select avg(age) from employees2;**

**select dept, avg(age) from employees2 group by dept;**

**select count(emp\_id), city from employees2**

**group by city**

**order by count(Emp\_Id) desc;**

**select year(Doj),count(emp\_id)**

**from employees2**

**group by year (Doj);**

* **LIMIT & OFFSET**

Select name,rankscore from movies limit 20;

Select name,rankscore from movies limit 20 offset 40; (starting na 40 entry pachina data avse)

* **Order By function:**

select \* from match1 order by batsman1runs desc limit 10;

select \* from match1 order by batsman1runs asc limit 10;

**Distinct Function:**

Select distinct first\_name, last\_name from directors order by first\_name; (**Distinct function thi unique value j male repeat na thay**)

* **Group By function:**

Syntax:

Select column\_names

From table\_name

Where condition

Group by column\_names

Order by column\_names

* **Having Function:**

select dept, avg(salary)

from employees2

group by dept

having avg(salary) > 53000;

* **Where Function:**

select \* from match1 where batsman1runs>5;

select \* from match1 where batsman1runs>5 order by batsman1runs desc limit 3;

* **Comparision operators ( =, != or <>, > , >=, <, <= )**

select \* from match1 where batsman1 = 'Virat kohli' order by batsman1runs desc limit 4;

* **V.Important**

**(= sign does not work with NULL)**

Select name, year, rankscore from movies where **rankscore is NULL** limit 10;

* **Logical Operators (AND, OR, NOT, ANY, BETWEEN, EXISTS, IN, LIKE, SOME)**

select name,year,rankscore from movies where rankscore>9 **AND** year>2000;

select name,year, rankscore from movies where **NOT** year<=2000;

select name, year, rankscore from movies where rankscore>9 **OR** year>2007;

SELECT NAME, YEAR, RANKSCOREFROM MOVIES **WHERE** YEAR BETWEEN 1999 AND 2000;

Select director\_id, genre from directors\_genres where genre IN (‘comedy’,’horror’);

**Same as genre = ‘comedy’ OR genre =’ horror’**

* Select name, year, rankscore from movies where name LIKE ‘Tis%’;

**( Means je name Tis thi start thase ej avse here % means 0 or more characters)**

* Select name, year, rankscore from movies where name LIKE ‘%tis%’;

**( Means je name nu end tis thi thase ej avse )**

* Select name, year, rankscore from movies where name LIKE ‘**Agn\_s%**’;

(Means vachhe khali ek j character ave ej avse )

* **V.important :**

**% is a wildcard character which represent 0 or more character.**

**Jo apdne % as a Symbol joie to percentage = ‘96\%’ aa rite use karvu**

**(\\_ escap character)**

**Aggregate function : COUNT, MIN, MAX, AVG, SUM**

Select min(year) from movies;

Select max(year) from movies;

Select count(\*) from movies;

Select count(\*) from movies where year>2000;

Select count(year) from movies;

**Group-By :**

# Find number of movies released per year

Select year, count(year) from movies group by year;

Select year, count(year) from movies group by year order by year;

Select year, count(year) year\_count from movies group by year order by year\_count;

**# year\_count is an alias**

**Having:**

Select year, count(year) year\_count from movies group by year having year\_count>1000;

**How to execute?**

1. Group by to create groups.

2. apply the aggregate function

3. Aplly having condition.

**# HAVING vs WHERE**

**## WHERE is applied on individual rows while having is applied on groups.**

**## HAVING is applied after grouping while WHERE is used before grouping.**

**## Having without group by is same as Where**

Select year, count(year) year\_count from movies where rankscore>9 group by year having year\_count>20;

**JOIN & NATURAL JOIN :**

**Select m.name, g.genre from movies m JOIN movies\_genres g ON m.id=g.movie\_id LIMIT20;**

(Here m is an alias for movies table and g is an alias for movies\_genres table)

**Natural JOIN: a join where we have the same column names across two tables**

**#T1: C1,C2**

**#T2: C1,C3,C4**

Select \* from T1 JOIN T2;

Select \* from T1 JOIN T2 using (c1);

**Inner and (LEFT,RIGHT & FULL)Outer joins :**

**Select m.name, g.genre from movies m LEFT OUTER JOIN movies\_genres g ON m.id=g.movie\_id LIMIT20;**

**3-WAY JOINs and K-Way joins:**

Select a.first\_name, a.last\_name from actors a JOIN roles r ON a.id=r.actor\_id JOIN movies m ON m.id=r.movie\_id AND m.name=’officer 444’;

**Sub-queries or Nested Queries or Inner Queries:**

Select first\_name, last\_name from actors id IN

( select actor\_id from roles WHERE movie\_id IN

(select id from movies where name=’Schindler’s list)

);

**Syntax:**

Select column\_name [, column\_name ]

FROM table1 [, table2 ]

WHERE column\_name OPERATOR

(select column\_name [, column\_name ]

FROM table1 [, table2 ]

[WHERE]);

**# first the inner query is executed and then other query is executed using the output values in the inner query.**

**# IN,NOT IN, EXISTS,NOT EXISTS , ANY, ALL, COMPARISION Operators :**

**Select \* FROM movies where rankscore >= (select MAX(rankscore) from movies);**

**Syntax:**

Select employee\_number, name

FROM employees emp

WHERE salary > (

Select AVG(salary)

FROM employees

WHERE department = emp.department))

**DML : Data Manipulation Language :**

**(SELECT, INSERT, UPDATE, DELETE)**

INSERT INTO movies(id, name, year, rankscore) values (412321, ‘Thor’, 2011, 7), (412322, ‘Iron Man’, 2008, 7.9);

**Copying rows from other tabled using nested query :**

INSER INTO phone\_book2

SELECT \*

FROM phone\_book

WHERE name IN (‘John’, ‘Peter’ )

UPDATE (column name) SET col1=val1, col2=val2 WHERE condition

**UPDATE movies SET rankscore=9 where id=412321;**

**DELETE FROM movies WHERE id=412321;**

To remove all rows: TRUNCATE TABLE Tablename;

**(TRUNCATE is part of DDL- Data Defination Language)**

**DDL ( Data Defination Language ):**

CREATE TABLE language ( id INT PRIMARY lang VARCHAR(50) NOT NULL);

NOT NULL : Ensure that a column cannot have a NULL value

UNIQUE : Ensure that all values in a column are different

PRIMARY Key : A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

Foreign Key : Uniquely identifies a row / record in another table

CHECK : Ensures that all values in a column satisfies a specific condition

DEFAULT : sets a default value for a column when no value is specified

INDEX : used to create and retrieve data from the database very quickly

**ALTER: ADD, MODIFY, DROP**

ALTER TABLE language ADD country varchar(50);

ALTER TABLE language MODIFY country varchar(60);

ALTER TABLE language DROP country;

**# Removes both table and all of the data permanently,**

DROP TABLE Tablename;

DROP TABLE Tablename IF EXISTS;

TRUNCATE TABLE Tablename;

**DCL ( Data Control Language ) :**

**For Data Base Admins :**

Who can access the data and who can’t

**GRANT**: to allow specified users to perform specified tasks.

**REVOKE**: to cancel previously granted or denied permissions.

GRANT ALL ON db1.\* TO ‘Jefrey’ @ ‘localhost’;

REVOKE ‘role1’ FROM ‘Janifier’@’localhost’;

REVOKE SELECT ON world.\* FROM ‘role3’;

(Here world is database and .\* means all the tables )