SQL - 2

SQL - 1 Notes

SQL Notes

JOINs:

#combine data in multiple tables

 # For each movie, print name and the genres
SELECT m.name, g.genre from movies m JOIN movies_genres g ON m.id=g.movie_id LIMIT 20;

 # table aliases: m and g

 # 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);

returns C1,C2,C3,C4
no need to use the keyword "ON"

 # Inner join (default) vs left outer vs right outer vs full-outer join.

T1: C1, C2, C3

SELECT m.name, g.genre from movies m LEFT JOIN movies_genres g ON m.id=g.movie_id LIMIT 20;

#LEFT JOIN or LEFT OUTER JOIN
#RIGHT JOIN or RIGHT OUTER JOIN
#FULL JOIN or FULL OUTER JOIN
#JOIN or INNER JOIN

NULL for missing counterpart rows.

 # 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';

#Practical note about joins: Joins can be expensive computationally when we have large tables.

****

Sub-Queries or Nested Queries or Inner Queries

List all actors in the movie Schindler's List
#https://www.imdb.com/title/tt0108052/fullcredits/?ref_=tt_ov_st_sm

```
SELECT first_name, last_name from actors WHERE 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 the outer query is executed using the output values in the inner query

IN, NOT IN, EXISTS, NOT EXISTS, ANY, ALL, Comparison operators

#EXISTS returns true if the subquery returns one or more records or NULL
ANY operator returns TRUE if any of the subquery values meet the condition.
ALL operators return TRUE if all of the subquery values meet the condition.

SELECT * FROM movies where rankscore >= ALL (SELECT MAX(rankscore) from movies);
get all movies whose rankscore is same as the maximum rankscore.

e.g: rankscore <> ALL(...)

https://en.wikipedia.org/wiki/Correlated_subquery

****

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), (412323, 'Iron Man 2', 2010, 7);

INSERT FROM one table to another using nested subquery:
https://en.wikipedia.org/wiki/Insert_(SQL)#Copying_rows_from_other_tables

****

UPDATE Command
UPDATE <TableName> SET col1=val1, col2=val2 WHERE condition

UPDATE movies SET rankscore=9 where id=412321;

Update multiple rows also.
Can be used along with Sub-queries.

****
#DELETE

DELETE FROM movies WHERE id=412321;

Remove all rows: TRUNCATE TABLE TableName;

****

Data Definition Language

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

Data Types: https://www.journaldev.com/16774/sql-data-types

Constraints: https://www.w3schools.com/sql/sql_constraints.asp

NOT NULL - Ensures that a column cannot have a NULL value
UNIQUE - Ensures 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 the table and all of the data permanently.
DROP TABLE Tablename;

DROP TABLE TableName IF EXISTS;

#https://dev.mysql.com/doc/refman/8.0/en/drop-table.html

TRUNCATE TABLE TableName;

as discussed earlier same as DELETE FROM TableName;

****

Data Control Language for DB Admins.

https://en.wikipedia.org/wiki/Data_control_language
https://dev.mysql.com/doc/refman/8.0/en/grant.htm
https://dev.mysql.com/doc/refman/8.0/en/revoke.html

SQL Problems

1. Employee dept wise avg salary

```
SELECT
  deptname department_name,
  ROUND(avg(salary), 2) average_salary
FROM employee
NATURAL JOIN department
GROUP BY deptid
```

2. Employee salary more than avg of their department

```
SELECT e.ename,
  e.sal,
  e.deptno
FROM emp e
WHERE e.sal > (
    SELECT AVG(sal)
    FROM emp
    WHERE emp.deptno = e.deptno
  )
```

References Shared

- https://sahilbansal17.github.io/eCSe-Notes/csl362/2019/01/24/CSL362.html
- https://dev.mysql.com/doc/refman/8.0/en/with.html
- https://www.educba.com/mysql-with/
- https://dev.mysql.com/doc/refman/5.6/en/explain-output.html
- https://en.wikipedia.org/wiki/Correlated_subquery
- https://dev.mysql.com/doc/refman/5.7/en/union.html
-