# SQL – Select

Select *<List of Columns and expressions (usually involving columns)>*

From *<List of Tables & Join Operators>*

Where *<List of Row conditions joined together by And, Or, Not>*

Group By *<list of grouping columns>*

Having *<list of group conditions connected by And, Or, Not >*

Order By *<list of sorting specifications>*

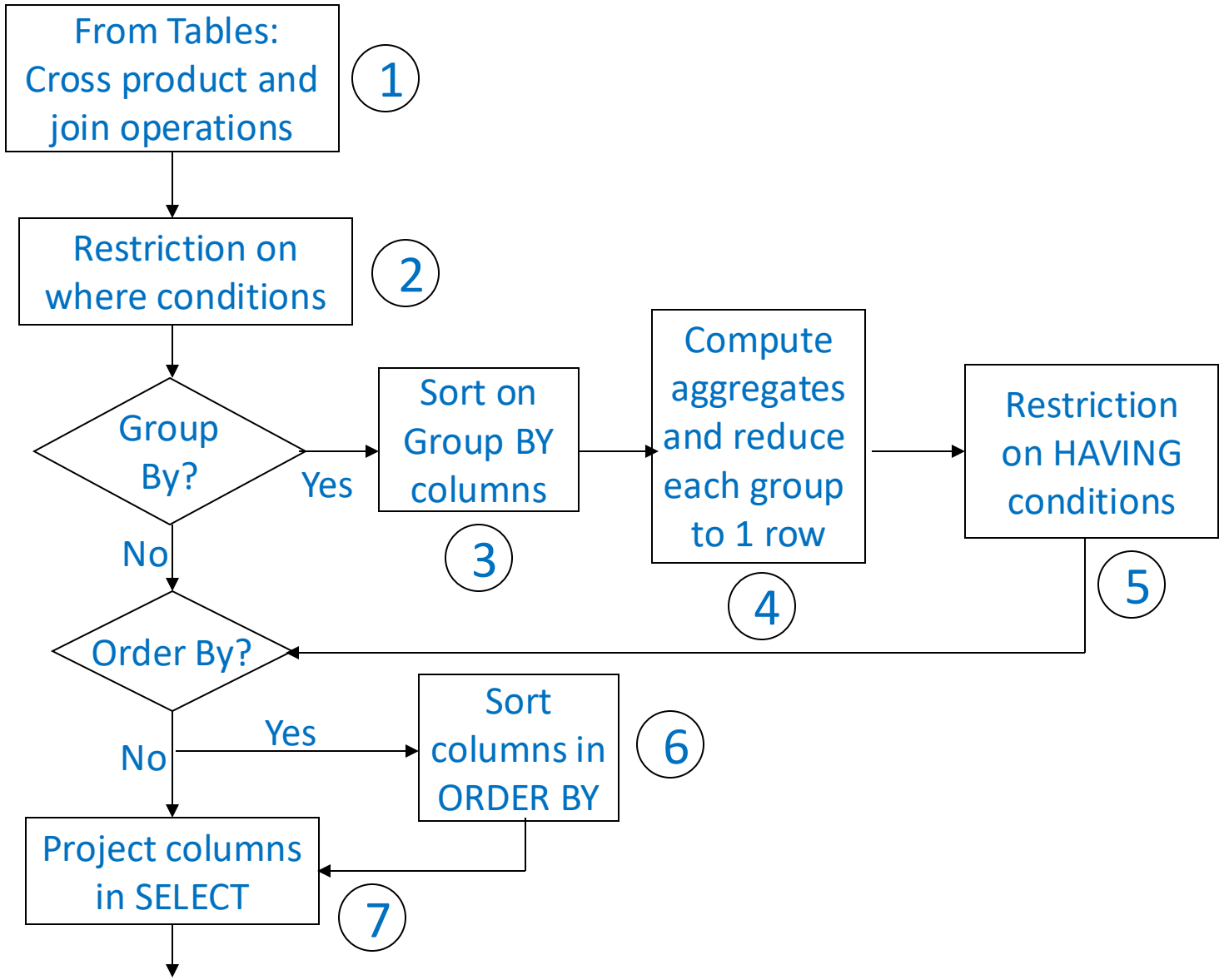
**Payroll**

| UserID | Name | Job | Salary |
|---|---|---|---|
| 123 | Jack | TA | 50000 |
| 345 | Allison | TA | 60000 |
| 567 | Magda | Prof | 120000 |
| 789 | Dan | Prof | 100000 |

| Name | UserID |
|---|---|
| Jack | 123 |
| Allison | 345 |

```
SELECT P.Name, P.UserID
  FROM Payroll AS P
 WHERE P.Job = 'TA';
```

# SQL – conceptual evaluation



# SQL – Order By

- Used to sort the results based on contents of a column
- Multiple levels of sort can be done by specifying multiple columns
- An expression can be used in Order By clause

**Syntax:**

```
Select function(column)
    From table1 [, table2 …]
    [Where condition]
    [Order By {Column | alias | position} [ASC | DESC]]
```

# SQL – Order By

**Example**: Sort Movies by profits in Ascending order

Select Movie_title, Gross, Budget, (Gross – Budget) as profits

From movies

Order BY profits

| Movie_title | Gross | Budget | Profit |
|---|---|---|---|
| Great Escape | 67.5 | 70 | -2.5 |
| Upside Down | 54 | 50 | 4 |
| Green Warrior | 96 | 80 | 16 |
| Blue Oranges | 28 | 7 | 21 |

# SQL – Group By

- Categorizes the query results according to the contents of a column in the database .
- Multiple levels of subgroups can be created by specifying multiple columns.

**Syntax:**

Select column1, [column2, …]

From table [, table …]

[Where condition]

Group By column1, [column2, ….]

Having [Condition]

Table: Customers

| customer_id | first_name | last_name | age | country |
|---|---|---|---|---|
| 1 | John | Doe | 31 | USA |
| 2 | Robert | Luna | 22 | USA |
| 3 | David | Robinson | 22 | UK |
| 4 | John | Reinhardt | 25 | UK |
| 5 | Betty | Doe | 28 | UAE |

```
SELECT country, COUNT(*) AS number
FROM Customers
GROUP BY country;
```

| country | number |
|---|---|
| UAE | 1 |
| UK | 2 |
| USA | 2 |

# SQL – Join

- A Join is a Query that combines data from multiple tables
  - Multiple tables are specified in the From Clause
  - For two tables to be joined in a sensible manner, they need to have data in common

**Example:**

Schema: Movies (movie_title, director_id, release_date)

People(person_fname, person_lname, person_id)

Query: Select movie_title, person_fname, person_lname

From Movies, People

Where director_id = person_id

# SQL – Joining Condition

For a useful Join query a joining condition is required

- Defined in where clause as relationships between columns
- Multiple conditions may be defined if multiple columns shared
- More than two tables can be joined in a query

**Example:** Find people who live in same state as studio

**Schema:**

Studios(studio_id, studio_state, studio_name, studio_city)

People(person_fname, person_lname, person_id, person_state, person_city)

**Query:**

```
Select person_fname, person_lname, studio_name
From Movies, People
Where studio_city = person_city
AND studio_state = person_state
```

# SQL – Self Join

Required to compare values within a single column

- Need to define aliases for the table names

**Example:** Find actors living in the same state

**Schema:**

People(person_fname, person_lname, person_id, person_state, person_city)

**Query:**

```
Select p1.person_id, p1.person_fname, p1.person_lname, p1.person_state
From People p1, People p2
Where p1.person_state = p2.person_state
AND p1.person_id != p2.person_id
```

**Note:**

Distinct operator is critical because if there are more than two people from any state each person will appear as many times as there are people from the state

# SQL – Left/Right Join

**Schema:**

People(person_fname, person_lname, person_id, person_state, person_city)

Movies(movie_id, movie_title, director_id, studio_id)

Location(movie_id, city, state)

**Query:**

```
Select movie_title, city, state
From Movies Left Join Locations
    On Movies.movie_id = Locations.movie_id
```

Includes all non matched movie titles

---

```
Select movie_title, person_fname, person_lname
From Movies Right Join People
    On Movies.director_id = Person.person_id
```

Includes all people not matching to directors