SELECT STATEMENT OVERVIEW

Topic 1: Statements and Clauses

Topic 2: Difference Between WHERE and HAVING

Topic 3: A Quick Look at GROUP BY

Topic 4: Execution Order of Queries

1. Statements and Clauses

Statements/Clauses	Definition
SELECT	To select data from a database, and store it in a result table
SELECT DISTINCT	To get rid of duplicate values in a column, and return only distinct (different) values
FROM	To list the table(s), and any joins required
JOIN	To combine rows from two or more tables
ON	To define the relationship between the tables to be combined
WHERE	To specify a condition; a filter
HAVING	To specify a condition, containing aggregate functions (COUNT, SUM, AVG, MIN, MAX)
GROUP BY	To group the result-set by column(s)
ORDER BY	To sort the result-set in ascending or descending order
TOP	To limit the result-set to a specified number of rows or percentage of rows

2. Difference Between WHERE and HAVING

*__WHERE__ applies to individual records, whereas __HAVING__ applies to summarized group records.

WHERE can be used with SELECT, UPDATE, DELETE. HAVING can be used with only SELECT statement.

WHERE is used to filter individual rows before aggregation, whereas HAVING is used to filter groups after aggregation.

__HAVING__ must follow a __GROUP BY__ clause. It filters the records of GROUP BY. Therefore, only the group(s) that meets the HAVING criteria will be returned.

Sample Database “Hunters”:

ID	NAME	AGE	SURNAME	SALARY
1	John	54	Winchester	2000.00
2	Dean	34	Winchester	10.00
3	Sam	30	Winchester	10.00
4	Cass	34	NULL	100.00
5	Bobby	57	Singer	8500.00
6	Mary	49	Winchester	4500.00
7	Emma	29	Stone	15000.00
8	Emma	28	Watson	20000.00
9	Emma	27	Roberts	15000.00

Example: List the hunters whose age is between 30 - 35

```
SELECT NAME, SURNAME, AGE
FROM Hunters
WHERE AGE BETWEEN 30 AND 35
```

Dean	Winchester	34
Sam	Winchester	30
Cass	NULL	34

It applies to every single record (row)

Example: List the hunters who have the same age

```
SELECT NAME, SURNAME, AGE
FROM Hunters
GROUP BY NAME, SURNAME, AGE
HAVING COUNT(AGE) >= 2
```

Dean	Winchester	34
Cass	NULL	34

It applies to the groups of the records.

Example: List the total salary of Winchesters

```
SELECT SURNAME, SUM(SALARY) AS TOTALSAL
FROM Hunters
WHERE SURNAME = 'Winchester'
GROUP BY SURNAME
```

```
SELECT SURNAME, SUM(SALARY) AS TOTALSAL
FROM Hunters
GROUP BY SURNAME
HAVING SURNAME = 'Winchester'
```

BOTH gives the same result?

SURNAME	TOTALSAL
Winchester	6520.00

3. A Quick Look at GROUP BY

Example: List the total salary, ordering by name

```
SELECT NAME, SUM(SALARY) AS TOTALSAL
FROM Hunters
GROUP BY NAME
```

Some hunters share the same name.

NAME	TOTALSAL
John	2000.00
Dean	10.00
Sam	10.00
Cass	100.00
Bobby	8500.00
Mary	4500.00
Emma	50000.00

Example: List the total salary, ordering by name and surname

```
SELECT NAME, SURNAME, SUM(SALARY) AS TOTALSAL
FROM Hunters
GROUP BY NAME, SURNAME
```

NAME	SURNAME	TOTALSAL
John	Winchester	2000.00
Dean	Winchester	10.00
Sam	Winchester	10.00
Cass	NULL	100.00
Bobby	Singer	8500.00
Mary	Winchester	4500.00
Emma	Stone	15000.00
Emma	Watson	20000.00
Emma	Roberts	15000.00

No hunter has the same name - surname double.

We list data for both NAME and SURNAME groups

Most Common Mistakes While Using GROUP BY,

Column ‘Table.Column' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

1. Missing GROUP BY clause

```
SELECT NAME, AVG(SURNAME) AS AVGSURNAME
FROM Hunters
```

2. Misuse of GROUP BY clause

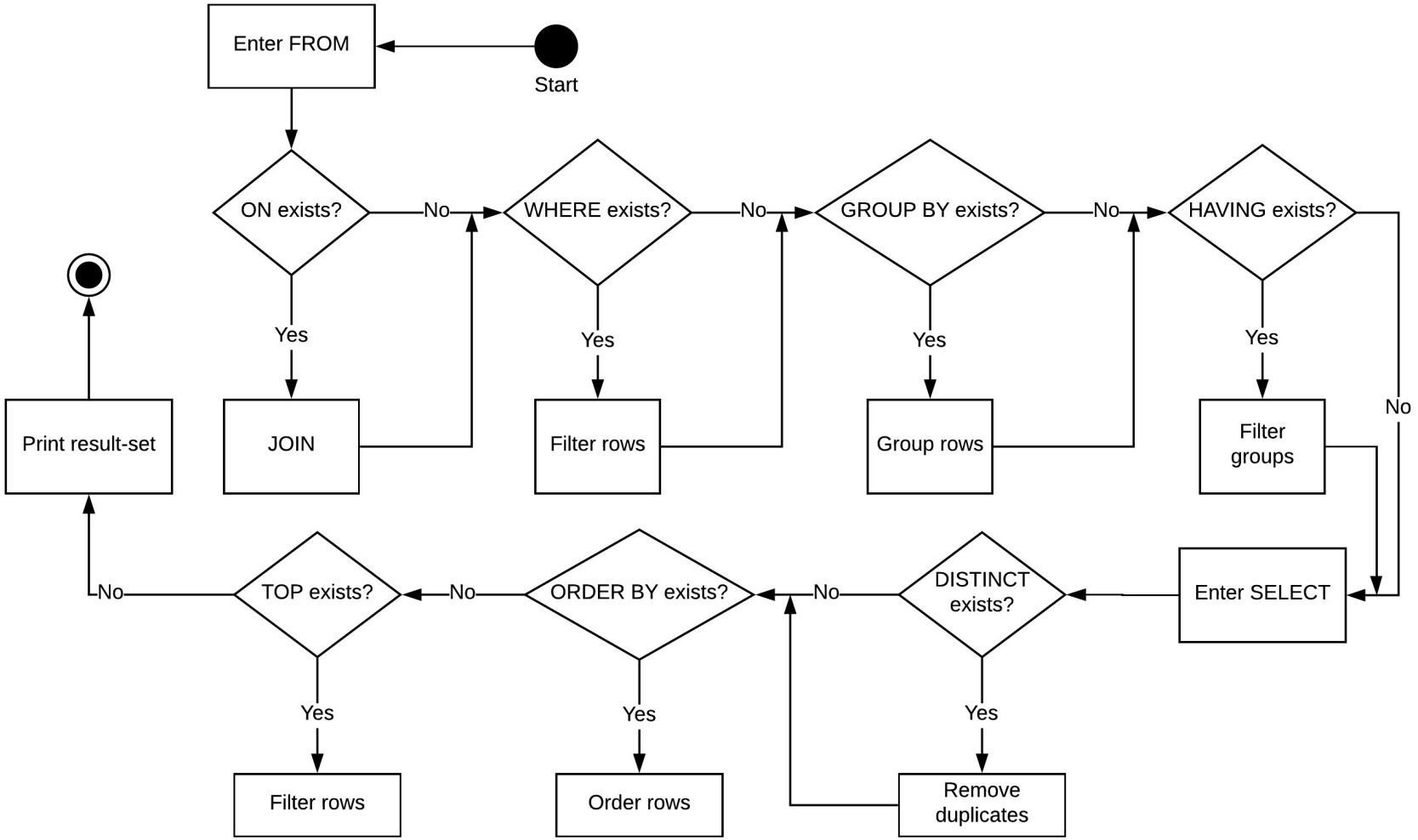
```
SELECT NAME, SURNAME, AVG(SALARY) AS AVGSAL
FROM Hunters
GROUP BY NAME
```

The column(s) in the SELECT statement must be part of either an aggregate function or the GROUP BY clause.

4. Execution Order of Queries in SQL Server

Know as “Logical Query Processing Phases”

full-diagram



Logical Query Processing Phases

(8) SELECT (9) DISTINCT (11) <TOP_specification> <select_list>
(1) FROM <left_table>
(3) JOIN <join_type> JOIN <right_table>
(2) ON <join_condition>
(4) WHERE <where_condition>
(5) GROUP BY <group_by_list>
(6) WITH {CUBE | ROLLUP}
(7) HAVING <having_condition>
(10) ORDERBY <order_by_list>

!! CUBE and ROLLUP operators are extensions of GROUP BY clause. Basically, they allow us to do several GROUP BY operations in on statement. They are used with WITH clause.

Example: Write an SQL query that returns top ten of the list of Customers who have at least 100 invoices from 1st April 2017 to YTD, from the highest to the lowest.
(Based on the Question3 from the obligatory exercises)

```
(7)SELECT (9)TOP 10 Customers.CustomerID,
CustomerName, COUNT(InvoiceID) AS InvoiceCount
(1)FROM Sales.Customers
(3)JOIN Sales.Invoices (2)ON
Sales.Invoices.CustomerID=Sales.Customers.CustomerID
(4)WHERE InvoiceDate >= '2017-04-01'
(5)GROUP BY Customers.CustomerID,
Customers.CustomerName
(6)HAVING COUNT(InvoiceID)>=100
(8)ORDER BY InvoiceCount DESC
```

	CustomerID	CustomerName	InvoiceCount
1	126	Tailspin Toys (Francis Mills, NJ)	119
2	457	Wingtip Toys (Portales, NM)	116
3	184	Tailspin Toys (South Euclid, OH)	114
4	902	Kamila Michnova	112
5	842	Biju Deb	111
6	901	Adrian Andreasson	111
7	547	Wingtip Toys (Chaseley, ND)	110
8	826	Leila Carvalho	110
9	1048	Abhra Ganguly	108
10	71	Tailspin Toys (Good Hart, MI)	107

Same Example with subquery

```
(10)SELECT TOP 10 SubQ.CustomerName, SubQ.InvoiceCount
(1)FROM ((8)SELECT Customers.CustomerID, CustomerName, COUNT(InvoiceID) AS InvoiceCount
(2)FROM Sales.Customers
(4)JOIN Sales.Invoices (3)ON Sales.Invoices.CustomerID=Sales.Customers.CustomerID
(5)WHERE InvoiceDate >= '2017-04-01'
(6)GROUP BY Customers.CustomerID, Customers.CustomerName
(7)HAVING COUNT(InvoiceID)>=100 ) (9)AS SubQ
(11)ORDER BY InvoiceCount DESC
```

	CustomerName	InvoiceCount
1	Tailspin Toys (Francis Mills, NJ)	119
2	Wingtip Toys (Portales, NM)	116
3	Tailspin Toys (South Euclid, OH)	114
4	Kamila Michnova	112
5	Biju Deb	111
6	Adrian Andreasson	111
7	Wingtip Toys (Chaseley, ND)	110
8	Leila Carvalho	110
9	Abhra Ganguly	108
10	Tailspin Toys (Good Hart, MI)	107

supplier_id	supplier_name	city	state
100	Microsoft	Redmond	Washington
200	Google	Mountain View	California
300	Oracle	Redwood City	California
400	Kimberly-Clark	Irving	Texas
500	Tyson Foods	Springdale	Arkansas
600	SC Johnson	Racine	Wisconsin
700	Dole Food Company	Westlake Village	California
800	Flowers Foods	Thomasville	Georgia
900	Electronic Arts	Redwood City	California

Example:

```
(2)SELECT (3)DISTINCT (5)TOP 2 state
(1)FROM suppliers
(4)ORDER BY state
```

STEP 1: No result-set yet. (Only FROM has executed)

STEP 2	**STEP 3**	**STEP 4**	**STEP 5**
(2)SELECT state **(1)**FROM suppliers	**(2)**SELECT **(3)**DISTINCT state **(1)**FROM suppliers	**(2)**SELECT **(3)**DISTINCT state **(1)**FROM suppliers **(4)**ORDER BY state	**(2)**SELECT **(3)**DISTINCT **(5)**TOP 2 state **(1)**FROM suppliers **(4)**ORDER BY state
state	state	state	state
Washington	Washington	Arkansas	Arkansas
California	California	California	California
California	Texas	Georgia	
Texas	Arkansas	Texas	
Arkansas	Wisconsin	Washington	
Wisconsin	Georgia	Wisconsin	
California			
Georgia			
California			

Q&A

THANKS

Atacan Demiralp
8407