



DELHI TECHNOLOGICAL UNIVERSITY

Name : Shrey

Roll Number : 2K19/MC/122

Subject : Database Management System Laboratory

Faculty : Prof. Aditya Kaushik

PRACTICAL - 6

AIM:

1. Use aggregate functions like max, min, count, sum.
2. Use GROUP BY and HAVING clause to fetch data from a group.
3. Check if any value exists in the result set using EXISTS and NOT EXISTS.

CODE AND OUTPUT:

Command to use MAX():

```
select MAX(Revenue) as largestRevenue from game_data;
```

Output :

The screenshot shows a SQL query execution interface. The query entered is `select MAX(Revenue) as largestRevenue from game_data;`. The result is displayed in a table with one row and one column, showing the value 993450000. The interface also includes a toolbar with options like Filter Rows, Export, and Wrap Cell Content. A message box at the bottom indicates that the query was executed successfully, returning 1 row(s) in 0.453 seconds.

largestRevenue
993450000

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result 1 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	16:08:50	select MAX(Revenue) as largestRevenue from game_data LIMIT 0, 1000	1 row(s) returned	0.453 sec / 0.000 sec

Command to use MIN():

```
select MIN(Revenue) as lowestRevenue from game_data;
```

Output :

The screenshot shows a SQL IDE interface with a query editor at the top containing the command: `select MIN(Revenue) as lowestRevenue from game_data;`. Below the editor, the 'Result Grid' displays a single row with the column name 'lowestRevenue' and the value '100000'. On the right side, a help message states: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.' At the bottom, the 'Output' pane shows a table of execution actions.

#	Time	Action	Message	Duration / Fetch
1	16:08:50	select MAX(Revenue) as largestRevenue from game_data LIMIT 0, 1000	1 row(s) returned	0.453 sec / 0.000 sec
2	16:15:31	select MIN(Revenue) as largestRevenue from game_data LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
3	16:17:20	select MIN(Revenue) as lowestRevenue from game_data LIMIT 0, 1000	1 row(s) returned	0.031 sec / 0.000 sec

Command to use COUNT():

```
select COUNT(Game_type) as Types_of_games from game_data;
```

Output :

The screenshot shows a SQL IDE interface with a query editor at the top containing the command: `select COUNT(Game_type) as Types_of_games from game_data;`. Below the editor, the 'Result Grid' displays a single row with the column name 'Types_of_games' and the value '14'. On the right side, a help message states: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.' At the bottom, the 'Output' pane shows a table of execution actions.

#	Time	Action	Message	Duration / Fetch
2	16:15:31	select MIN(Revenue) as largestRevenue from game_data LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
3	16:17:20	select MIN(Revenue) as lowestRevenue from game_data LIMIT 0, 1000	1 row(s) returned	0.031 sec / 0.000 sec
4	16:18:07	select COUNT(Game_type) as Types_of_games from game_data LIMIT 0, 1000	1 row(s) returned	0.047 sec / 0.000 sec

Command to use SUM():

```
select SUM(Revenue) as Total_revenue from game_data;
```

Output :

The screenshot shows a SQL IDE interface with a query editor at the top containing the command: `select SUM(Revenue) as Total_revenue from game_data;`. Below the editor, the 'Result Grid' displays a single row with the column 'Total_revenue' and the value '3225571829'. The 'Output' pane at the bottom shows a log of actions, including the execution of the SUM query at 16:19:25, which returned 1 row(s) in 0.016 seconds.

game	game_type
Fortnite	Action
Assasins Creed	Action
Cyberpunk	Knowledge
doombag	adventure
Eternal War	Arcade
Footgoals	Sports
Genshin	Open World
teenpatti	Strategy
tekken	Action
Shoot Duck	Shooting
Super Mario	Open World
Diablo	Arcade
Resident Evil	Survival

Command to use GROUP BY:

```
SELECT game,game_type  
FROM game_data  
WHERE Revenue>901833  
GROUP BY game;
```

Output :

The screenshot shows a SQL IDE interface with a query editor at the top containing the command: `SELECT game,game_type FROM game_data WHERE Revenue>901833 GROUP BY game;`. Below the editor, the 'Result Grid' displays a list of games and their types. The 'Output' pane at the bottom shows a log of actions, including the execution of the GROUP BY query at 16:25:49, which returned 14 row(s) in 0.313 seconds.

game	game_type
Fortnite	Action
Assasins Creed	Action
Cyberpunk	Knowledge
doombag	adventure
Eternal War	Arcade
Footgoals	Sports
Genshin	Open World
teenpatti	Strategy
tekken	Action
Shoot Duck	Shooting
Super Mario	Open World
Diablo	Arcade
Resident Evil	Survival

Command to use GROUP BY and HAVING:

```
CREATE VIEW top_action_games AS
SELECT game,game_type
FROM game_data
WHERE Revenue>901833
GROUP BY game
Having game_type = "Action";
SELECT * from top_action_games;
```

Output :

The screenshot displays the SQL Enterprise Manager interface. The main window shows a SQL script with the following commands:

```
1 CREATE VIEW top_action_games AS
2 SELECT game,game_type
3 FROM game_data
4 WHERE Revenue>901833
5 GROUP BY game
6 Having game_type = "Action";
7
8 SELECT * from top_action_games
```

Below the script, the 'Result Grid' shows the output of the queries:

game	game_type
Fortnite	Action
Assasins Creed	Action
tekken	Action

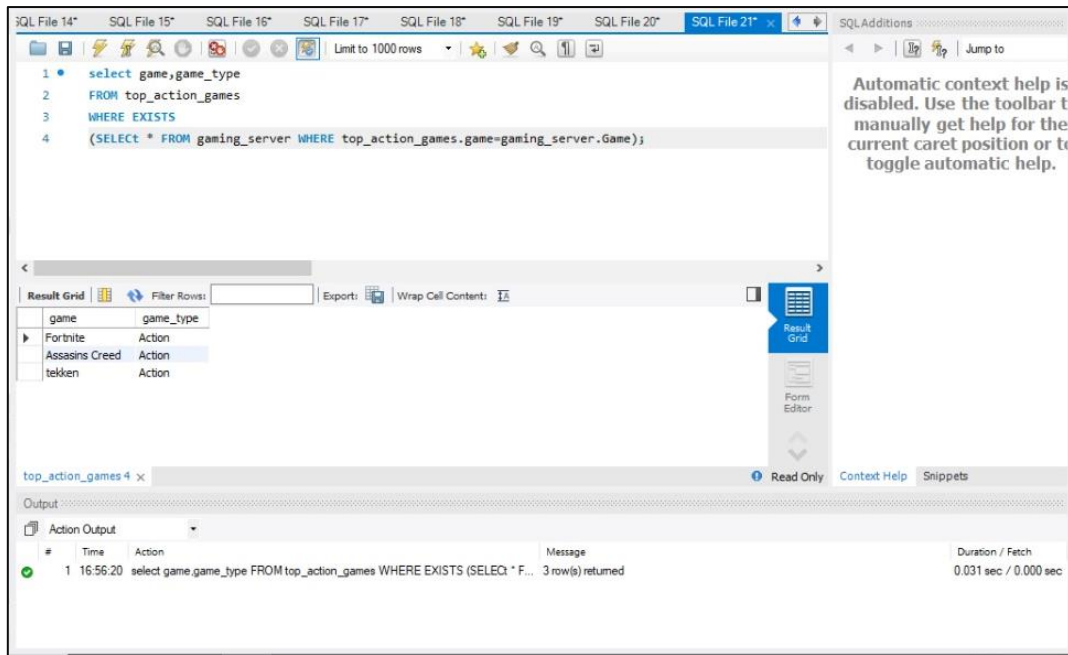
The bottom pane shows the 'Output' window with the following log:

#	Time	Action	Message	Duration / Fetch
1	16:30:00	SELECT game,game_type FROM game_data WHERE Revenue>901833 GRO...	3 row(s) returned	0.000 sec / 0.000 sec
2	16:45:34	CREATE VIEW top_action_games AS SELECT game,game_type FROM game_...	0 row(s) affected	1.797 sec
3	16:46:06	SELECT * from top_action_games LIMIT 0, 1000	3 row(s) returned	0.156 sec / 0.000 sec

Command to use EXISTS:

```
select game,game_type
FROM top_action_games
WHERE EXISTS
(SELECT * FROM gaming_server WHERE top_action_games.game=gaming_server.Game);
```

Output :



Command to use NOT EXISTS:

```

select game,game_type
FROM top_action_games
WHERE NOT EXISTS
(SELECT * FROM gaming_server WHERE top_action_games.game=gaming_server.Game);

```

Output :

