

# ASSIGNMENT-5

## 1) Total aggregations using SQL

The SQL aggregate functions — AVG, COUNT, DISTINCT, MAX, MIN, SUM — all return a value computed or derived from one column's values, after discarding any NULL values.

- 1) **AVG()**: The AVG() function returns the average value of a numeric column.

### Retrieve average price of all products

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure with tables: career2, currental, data, db, dd, demo, joins, and com. The 'products' table is selected.
- SQL Editor:** Contains the following SQL code:

```
-- TOTAL AGGREGATIONS USING SQL
-- 1) AVG()
-- Retrieve average price of all products
select AVG(price) as AVG_PRICE from products;
```
- Result Grid:** Displays the result of the query: AVG\_PRICE = 377.
- Action Output:** Shows the execution log with three entries:

#	Time	Action	Message	Duration / Fetch
174	23:49:48	SELECT productID, name FROM products EXCEPT SELECT productID, * FROM Orderitem	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'EXCEPT SELECT productID, * FROM Orderitem' at line 1	0.000 sec / 0.000 sec
175	23:49:48	SELECT productID, name FROM products UNION SELECT productID, * FROM Orderitem	17 row(s) returned	0.000 sec / 0.000 sec
176	23:49:48	select AVG(price) as AVG_PRICE from products LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

- 2) **COUNT()**: The COUNT() function returns the number of rows that matches a specified criterion.

### Retrieve total count of orders

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure with tables: career2, currental, data, db, dd, demo, joins, and com. The 'orderitem' table is selected.
- SQL Editor:** Contains the following SQL code:

```
select AVG(price) as AVG_PRICE from products;
-- 2) COUNT()
-- Retrieve total count of orders
select COUNT(*) as Order_count from orderitem;
select * from orderitem;
```
- Result Grid:** Displays the result of the query: Order\_count = 9.
- Action Output:** Shows the execution log with three entries:

#	Time	Action	Message	Duration / Fetch
181	23:49:04	selected max(stockQuantity) as MAX_QUANTITY from products LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
182	23:49:23	selected sum(price) as totalAmount from products LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
183	23:49:49	select COUNT(*) as Order_count from orderitem LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

### 3) MIN(): The MIN() function returns the smallest value of the selected column

Retrieve minimum item amount of an order

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL57
- SQL Editor:** Contains the following SQL code:

```
-- 3) MIN()
-- Retrieve minimum item amount of an order
select min(itemAmount) as Min_Amount from orderitem;
```
- Result Grid:** Shows the result of the query: Min\_Amount = 50.
- Action Output:** Displays the log of actions taken by the system, including the execution of the query.
- System Bar:** Shows the date and time as 22-01-2024 23:51.

### 4) MAX(): The MAX() function returns the largest value of the selected column.

Retrieve maximum quantity of a product

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL57
- SQL Editor:** Contains the following SQL code:

```
-- 4) MAX()
-- Retrieve maximum quantity of a product
select max(stockQuantity) as MAX_QUANTITY
from products;
```
- Result Grid:** Shows the result of the query: MAX\_QUANTITY = 30.
- Action Output:** Displays the log of actions taken by the system, including the execution of the query.
- System Bar:** Shows the date and time as 22-01-2024 23:54.

## 5) SUM() : The SUM() function returns the total sum of a numeric column.

Retrieve total price of all products

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL57
- Query Editor:** SQL File 5\* (subqueries & total aggregations\*)  
The code is:

```
205      from products;
206
207
208
209
210
211 -- 5) SUM()
212 -- Retrieve total price of all products
213 • select sum(price) as totalAmount from products;
```
- Result Grid:** Shows the result of the query: totalAmount = 3770
- Action Output:** Shows the execution log with three entries:184 23:51:56 select min(itemAmount) as Min\_Amount from orderitem LIMIT 0, 1000
1 row(s) returned
0.000 sec / 0.000 sec
185 23:51:39 select max(stockQuantity) as MAX\_QUANTITY from products LIMIT 0, 1000
1 row(s) returned
0.000 sec / 0.000 sec
186 23:56:23 select sum(price) as totalAmount from products LIMIT 0, 1000
1 row(s) returned
0.000 sec / 0.000 sec

## 6) DISTINCT():

Retrieve names of distinct products.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL57
- Query Editor:** SQL File 5\* (subqueries & total aggregations\*)  
The code is:

```
203
204
205
206 -- DISTINCT
207 -- Retrieve names of distinct products
208 • select distinct name from products;
```
- Result Grid:** Shows the distinct product names:

name
Smartphone
Tablet
Headphones
TV
Coffee Maker
Refrigerator
Microwave Oven
Blender
Vacuum Cleaner
- Action Output:** Shows the execution log with three entries:184 23:51:56 select min(itemAmount) as Min\_Amount from orderitem LIMIT 0, 1000
1 row(s) returned
0.000 sec / 0.000 sec
185 23:51:39 select max(stockQuantity) as MAX\_QUANTITY from products LIMIT 0, 1000
1 row(s) returned
0.000 sec / 0.000 sec
186 23:56:23 select sum(price) as totalAmount from products LIMIT 0, 1000
1 row(s) returned
0.000 sec / 0.000 sec

## 2) Order of Execution of SQL Queries:

- The order of execution determines how the database processes your query and retrieves the desired results.
- SQL order of execution refers to **the sequence in which different clauses and operations within a SQL query are processed** by the database management system.
- **Order:** FROM , WHERE , GROUP BY , HAVING , SELECT , ORDER BY, LIMIT/OFFSET.

### 1) Example:

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
211
212 •  SELECT name, AVG(price) AS avg_price
213   FROM products
214   WHERE stockquantity > 10
215   GROUP BY name
216   HAVING AVG(price) > 50
217   ORDER BY avg_price DESC
218   LIMIT 5;
219
```

The result grid shows the following data:

name	avg_price
Smartphone	600
Tablet	300
Headphones	150
Microwave Oven	80
Blender	70

The status bar at the bottom right indicates the date and time: 23-01-2024 00:16.

### 2) Retrieve the names of products that have been ordered by customers whose last names start with the letter 'h'

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
219
220
221 -- retrieve the names of products that have been ordered by customers whose last names start
222 -- with the letter 'h'
223
224 • select p.name
225   from products p
226   join orderitem o on p.productid=o.productid
227   join customers c on o.customerid=c.customerid
228   where c.lastname like 'h%'
229   group by p.name
230   having count(distinct o.customerid)>0;
231
```

The result grid shows the following data:

name
Smartphone
Tablet
Headphones
Microwave Oven
Blender

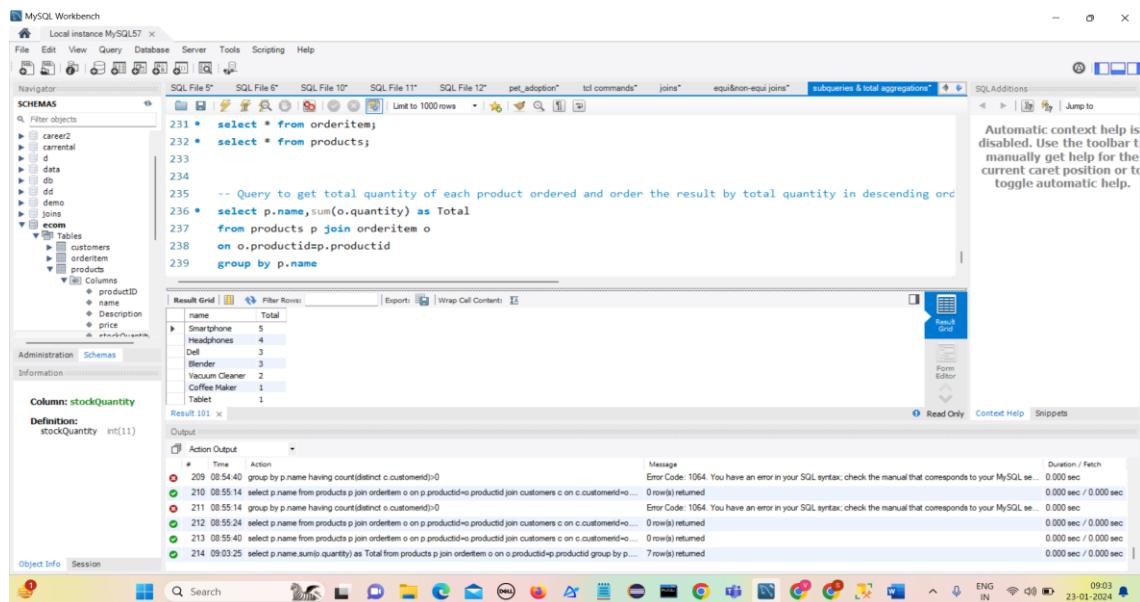
The status bar at the bottom right indicates the date and time: 23-01-2024 08:56.

### 3) Rules and Restrictions to Group and Filter Data in SQL queries

#### 1) GROUP BY

GROUP BY enables you to use aggregate functions on groups of data returned from a query.

**Query to get total quantity of each product ordered and order the result by total quantity in descending order.**



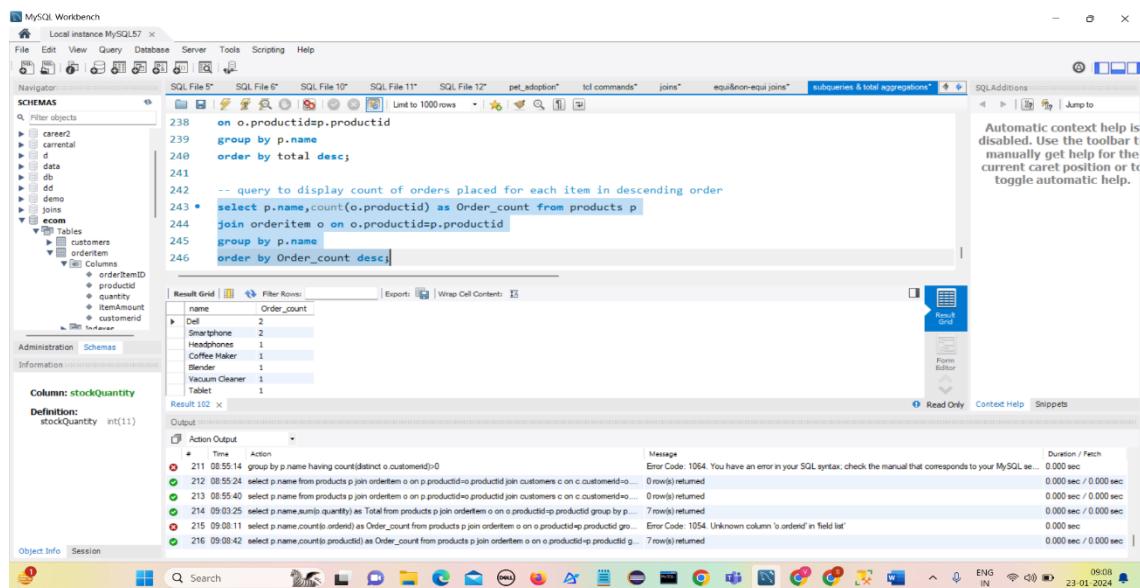
The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the following SQL code:

```
231 * select * from orderitem;
232 * select * from products;
233
234
235 -- Query to get total quantity of each product ordered and order the result by total quantity in descending order
236 * select p.name,sum(o.quantity) as Total
237   from products p join orderitem o
238     on o.productid=p.productid
239   group by p.name
```
- Result Grid:** Displays the results of the query:

name	Total
Smartphone	5
Headphones	4
Dell	3
Blender	3
Vacuum Cleaner	2
Coffee Maker	1
Tablet	1
- Action Output:** Shows the history of actions taken on the query, including errors and warnings.

**Query to display count of orders placed for each item in descending order**



The screenshot shows the MySQL Workbench interface with the following details:

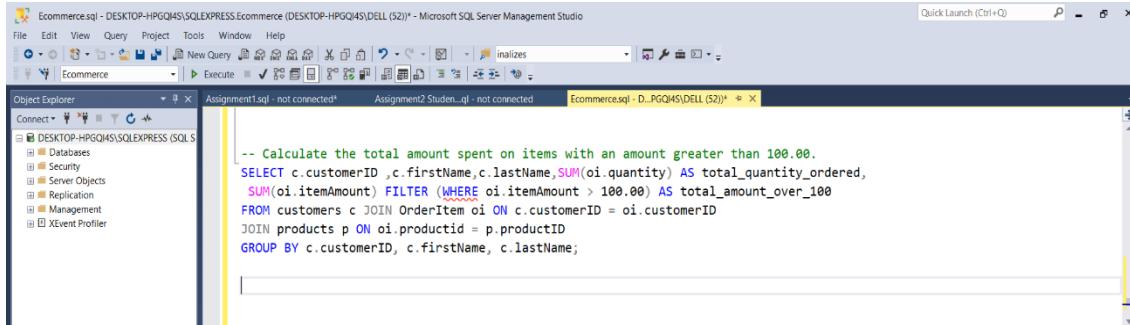
- SQL Editor:** Contains the following SQL code:

```
238 on o.productid=p.productid
239 group by p.name
240 order by total desc;
241
242 -- query to display count of orders placed for each item in descending order
243 * select p.name,count(o.productid) as Order_count from products p
244 join orderitem o on o.productid=p.productid
245 group by p.name
246 order by Order_count desc;
```
- Result Grid:** Displays the results of the query:

name	Order_count
Dell	2
Smartphone	2
Headphones	1
Coffee Maker	1
Blender	1
Vacuum Cleaner	1
Tablet	1
- Action Output:** Shows the history of actions taken on the query, including errors and warnings.

## 2) FILTER:

FILTER is a modifier used on an aggregate function to limit the values used in an aggregation.

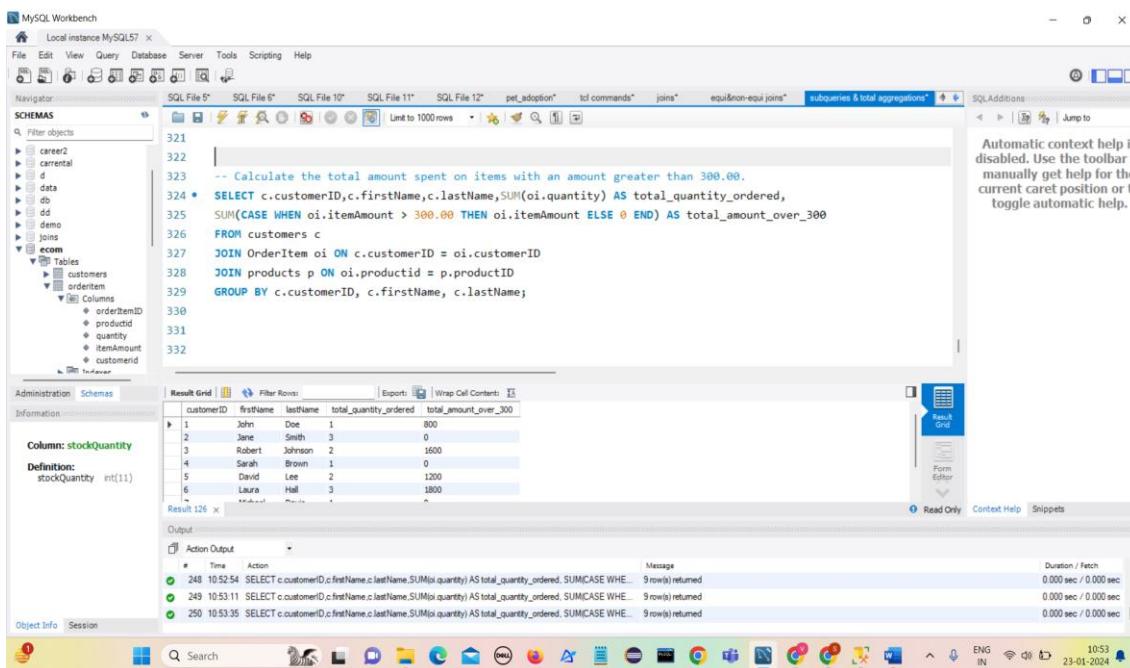


```
-- Calculate the total amount spent on items with an amount greater than 100.00.
SELECT c.customerID ,c.firstName,c.lastName,SUM(oi.quantity) AS total_quantity_ordered,
SUM(oi.itemAmount) FILTER (WHERE oi.itemAmount > 100.00) AS total_amount_over_100
FROM customers c JOIN OrderItem oi ON c.customerID = oi.customerID
JOIN products p ON oi.productid = p.productID
GROUP BY c.customerID, c.firstName, c.lastName;
```

It is better to use case instead of filter.

## 3) CASE statement

**Calculate the total amount spent on items with an amount greater than 300.00.**



```
-- Calculate the total amount spent on items with an amount greater than 300.00.
SELECT c.customerID,c.firstName,c.lastName,SUM(oi.quantity) AS total_quantity_ordered,
SUM(CASE WHEN oi.itemAmount > 300.00 THEN oi.itemAmount ELSE 0 END) AS total_amount_over_300
FROM customers c
JOIN OrderItem oi ON c.customerID = oi.customerID
JOIN products p ON oi.productid = p.productID
GROUP BY c.customerID, c.firstName, c.lastName;
```

customerID	firstName	lastName	total_quantity_ordered	total_amount_over_300
1	John	Doe	1	800
2	Jane	Smith	3	0
3	Robert	Johnson	2	1600
4	Sarah	Brown	1	0
5	David	Lee	2	1200
6	Laura	Hall	3	1800

## 4) FILTERING DATA USING SQL QUERIES

### 1) Using where clause

The screenshot shows the MySQL Workbench interface with a query editor window. The code is as follows:

```
297 GROUP BY c.customerID, c.firstName, c.lastName;
298
299
300 * select * from customers;
301 -- using where clause
302 * select * from customers
303 where email= 'johndoe@example.com' or customerid=1;
304
305
306
307
308
```

The result grid shows one row of data:

customerID	firstName	lastName	Email	address
1	John	Doe	johndoe@example.com	123 Main St, City

The output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
10	22:56:58	select * from customers LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
11	22:57:41	select * from customers where email= 'johndoe@example.com' and customerid=1 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
12	22:57:51	select * from customers where email= 'johndoe@example.com' or customerid=1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

### 2) Using group by and having clause

The screenshot shows the MySQL Workbench interface with a query editor window. The code is as follows:

```
303 where email= 'johndoe@example.com' or customerid=1;
304
305 -- Using group by and having clause
306 * select name, AVG(price) as avg_price from products
307 where stockQuantity > 10
308 group by name
309 having avg(price) > 50;
```

The result grid shows the following data:

name	avg_price
Blender	70
Headphones	150
Microwave Oven	80
Smartphone	600
Tablet	300

The output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
11	22:57:41	select * from customers where email= 'johndoe@example.com' and customerid=1 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
12	22:57:51	select * from customers where email= 'johndoe@example.com' or customerid=1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
13	23:01:52	select name, AVG(price) as avg_price from products where stockQuantity > 10 group by name having avg(price) > 50	5 row(s) returned	0.000 sec / 0.000 sec

## 5) Group by aggregations using sql

Count of orders of each product

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
310
311 -- group by aggregations using sql queries
312 select * from orderitem;
313
314 • select p.name, count(o.productid) as Order_count from orderitem o
315 join products p
316 on p.productid=o.productid
317 group by p.name;
318
319
```

The result grid shows the following data:

name	Order_count
Blender	1
Coffee Maker	1
Headphones	1
Smartphone	2
Tablet	1
Vacuum Cleaner	1
Dell	2

The output pane shows the execution log:

Action	Time	Message	Duration / Fetch
select * from orderitem	15 23:05:49	LIMIT 0, 1000 rows returned	0.000 sec / 0.000 sec
select p.name,count(o.productid) from orderitem o join products p on p.productid=o.productid group by p.name;	16 23:05:20	7 row(s) returned	0.015 sec / 0.000 sec
select p.name,count(o.productid) as Order_count from orderitem o join products p on p.productid=o.productid group by p.name;	17 23:05:34	7 row(s) returned	0.000 sec / 0.000 sec

## 6) Filter data on aggregated results using group by and having

Product names having stock quantity more than 10 and average price greater than 50.

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
314 • select p.name, count(o.productid) as Order_count from orderitem o
315 join products p
316 on p.productid=o.productid
317 group by p.name;
318
319 -- Filter data on Aggregated results using group by and having
320 • select name, AVG(price) as avg_price from products
321 where stockquantity > 10
322 group by name
323 having avg(price) > 50;
```

The result grid shows the following data:

name	avg_price
Blender	70
Headphones	150
Microwave Oven	80
Smartphone	600
Tablet	300

The output pane shows the execution log:

Action	Time	Message	Duration / Fetch
select p.name,count(o.productid) from orderitem o join products p on p.productid=o.productid group by p.name;	16 23:08:20	7 row(s) returned	0.015 sec / 0.000 sec
select p.name,count(o.productid) as Order_count from orderitem o join products p on p.productid=o.productid group by p.name;	17 23:08:34	7 row(s) returned	0.000 sec / 0.000 sec
select name, AVG(price) as avg_price from products where stockquantity > 10 group by name having avg(price) > 50;	18 23:12:01	5 row(s) returned	0.000 sec / 0.000 sec

## String functions:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, a query window displays several SQL statements under the heading '--String Functions'. The results pane below shows the output for each function. The system status bar at the bottom indicates the query was executed successfully.

```
--String Functions
select ascii('KM') as ASCII ; -- return leftmost ascii value
select char(69) as ASCII_Value; -- return ascii value to character
select len('Welcome to MySQL') as Length;-- return length
select lower('SRIGANGA') as Lower_Case;-- convert to lowercase
select replace('Sri Ganga','Sri','ganga');-- replace
select reverse('Nizamabad') as Rev;-- reverse the string
select upper('shivathmika') as Upper_case;-- converts to upper
select str(120.378,8,4) as Output;-- STR(number, length, decimals)|
```

ASCII	75
ASCII_Value	E
Length	16
Lower_Case	[No column name]
	ganga Ganga
Rev	dabamazN
Upper_case	SHIVATHMIIKA
Output	120.3780

Query executed successfully.

- 1) **Ascii-** Returns the ASCII value for the specific character
- 2) **char-** Returns the character based on the ASCII code.
- 3) **len-** Returns the length of a string
- 4) **lower-** Converts a string to lower-case
- 5) **replace-** The REPLACE() function replaces all occurrences of a substring within a string, with a new substring.  
REPLACE(string, old\_string, new\_string)
- 6) **reverse-** Reverses a string and returns the result
- 7) **upper-** Converts a string to upper-case
- 8) **str-** Returns a number as string

## Mathematical Functions:

- 1) **abs-** returns absolute value
- 2) **sin-** returns angle in radians
- 3) **ceiling-** returns the smallest or greater to the specified value
- 4) **exp-** returns the exponential value
- 5) **floor-** returns floor value
- 6) **log-** return logarithmic value

```

Ecommerce.sql - DESKTOP-HPGQ4S\SQLEXPRESS.Ecommerce (DESKTOP-HPGQ4S\DELL (52)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Object Explorer Connect > Ecommerce | Execute New Query Task List Find Replace Properties
Assignment1.sql - not connected Assignment2.Studen..ql - not connected Ecommerce.sql - D...PGQ4S\DELL (52)*
select year ('2023-05-3');--return year value

--Mathematical Functions
select abs(-101) as Absolute;--returns absolute value
select sin(1.5) as Sine;--returns angle in radians
select ceiling(14.01) as Cell;--returns the smallest or greater to the specified value
select exp(4.5) as Exponential;--returns the exponential value
select floor(14.75) as Floor_value;--returns floor value
select log(5.4) as Logarithmic;--return logarithmic value

Results [#] Messages
Absolute
1 101

Sine
1 0.997494808604054

Cell
1 15

Exponential
1 90.0171313005218

Floor_value
1 14

Logarithmic
1 1.6983995357023

Query executed successfully.
Ln 239 Col 58 Ch 58 INS
DESKTOP-HPGQ4S\SQLEXPRESS .. DESKTOP-HPGQ4S\DELL (52) Ecommerce 00:00:00 6 rows
Ready Search
23-01-2024 10:30 ENG IN

```

## Date Functions:

- 1) **getdate**- get current date and time
- 2) **dateadd**- add months to existed date
- 3) **datediff**- it will return the difference of date, months, years also  
DATEDIFF(*interval*, *date1*, *date2*)
- 4) **datepart**- return months value.
- 5) **day**- return value of date of that particular day
- 6) **month**- return month value
- 7) **year**- return year value

```

Ecommerce.sql - DESKTOP-HPGQ4S\SQLEXPRESS.Ecommerce (DESKTOP-HPGQ4S\DELL (52)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Object Explorer Connect > Ecommerce | Execute New Query Task List Find Replace Properties
Assignment1.sql - not connected Assignment2.Studen..ql - not connected Ecommerce.sql - D...PGQ4S\DELL (52)*
-- DATE FUNCTIONS
select getdate ()--get current date and time
select dateadd (mm, 2, '2022-06-12')--add months to existed date
select datediff ( year, convert (datetime, '2002-09-16'), convert ( datetime, '2008-06-06'))--it will return the difference of date,months,years also
select datepart (mm, '2009-7-2')--return months value
select day ('2024-01-06')--return value of date of that particular day
select month ('2024-07-31')--return month value
select year ('2023-05-3')--return year value

Results [#] Messages
(No column name)
1 2024-01-23 10:33:27.817

(No column name)
1 2022-08-12 00:00:00.000

(No column name)
1 8

(No column name)
1 7

(No column name)
1 6

(No column name)
1 7

(No column name)
1 2023

Query executed successfully.
Ln 235 Col 47 Ch 47 INS
DESKTOP-HPGQ4S\SQLEXPRESS .. DESKTOP-HPGQ4S\DELL (52) Ecommerce 00:00:00 7 rows
Ready Search
23-01-2024 10:33 ENG IN

```

## System Functions

- 1) **host\_name**-return system hostname
- 2) **Host\_ID**-return host id
- 3) **SUSER\_ID**-Returns the login identification number of the user
- 4) **USER\_ID**-Returns the identification number for a database user
- 5) **DB\_NAME** -Returns database name

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the following T-SQL code:

```
--System functions
select host_name() as Host_name;--return system hostname
select Host_ID() as HostID;--return host id
select SUSER_ID() as SuserID;--Returns the login identification number of the user
select USER_ID() as UserID;-- Returns the identification number for a database user
select DB_NAME() as DatabaseName;-- return database name
```

The results pane shows the output for each function:

Host_name	HostID	SuserID	UserID	DatabaseName
DESKTOP-HPGQI4S	9032	256	1	datclean

A status bar at the bottom indicates "Query executed successfully." and "5 rows".

## TCL Commands

Create table employee and insert values into it.

Retrieve the records of employee table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer pane on the left shows a connection to 'DESKTOP-HPGQH45\SQLEXPRESS'. The main query window contains the following TCL commands:

```
-- Create table employee
(
    id int primary key,
    name varchar(20),
    age int,
    email varchar(20)
);
sp_help employee; -- displays structure of table
-- Insert records into employee table
insert into employee(id,name,age,email) values(101,'alex',30,'alex@gmail.com');
insert into employee(id,name,age,email) values(102,'bob',20,'bob@gmail.com');
insert into employee(id,name,age,email) values(103,'sunny',40,'sunny@gmail.com');
insert into employee(id,name,age,email) values(104,'stella',35,'stella@gmail.com');
insert into employee(id,name,age,email) values(105,'nani',40,'nani@gmail.com');
insert into employee(id,name,age,email) values(106,'stella',35,'stella@gmail.com');
insert into employee(id,name,age,email) values(107,'nani',35,'nani@gmail.com');
select * from employee;
```

The results pane below shows the inserted data:

	id	name	age	email
1	101	alex	30	alex@gmail.com
2	102	bob	20	bob@gmail.com
3	103	sunny	40	sunny@gmail.com
4	104	stella	35	stella@gmail.com
5	105	nani	40	nani@gmail.com
6	106	stella	35	stella@gmail.com
7	107	nani	35	nani@gmail.com

At the bottom, the status bar indicates 'Query executed successfully.'

Add column salary and update the values of salary.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer pane on the left shows a connection to 'DESKTOP-HPGQH45\SQLEXPRESS'. The main query window contains the following TCL commands:

```
-- distinct
select distinct name from employee;

-- add salary column
alter table employee add salary int;
select * from employee;

-- update salary values
update employee set salary=20000 where name='alex';
update employee set salary=30000 where name='bob';
update employee set salary=35000 where name='sunny';
update employee set salary=35000 where name='stella';
update employee set salary=50000 where name='nani';

select * from employee;
```

The results pane below shows the updated data:

	id	name	age	email	salary
1	101	alex	30	alex@gmail.com	20000
2	102	bob	20	bob@gmail.com	30000
3	103	sunny	40	sunny@gmail.com	35000
4	104	stella	35	stella@gmail.com	35000
5	105	nani	40	nani@gmail.com	50000
6	106	stella	35	stella@gmail.com	35000
7	107	nani	35	nani@gmail.com	50000

At the bottom, the status bar indicates 'Query executed successfully.'

Add another column department and update values.

```
datacleaning & TC - DESKTOP-HPGQH4S\SQLEXPRESS.dataclean (DESKTOP-HPGQH4S\DELL (61)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Object Explorer
Connect Connect ...
DESKTOP-HPGQH4S\SQLEXPRESS (SQL Server)
Databases Security
Server Objects Replication Management XEvent Profiler
dataclean
New Query Execute
datacleaning & TC - DESKTOP-HPGQH4S\DELL (61) * X
update employee set salary=20000 where name='alex';
update employee set salary=30000 where name='bob';
update employee set salary=35000 where name='sunny';
update employee set salary=35000 where name='stella';
update employee set salary=50000 where name='nani';

select * from employee;

-- add department column
alter table employee add department varchar(30);
--update department values
update employee set department='sales' where name='stella';
update employee set department='HR' where name='nani';
update employee set department='IT' where name='alex';
update employee set department='salesforce' where name='bob';
update employee set department='HR' where name='sunny';

select * from employee;
```

Results Messages

	id	name	age	email	salary	department
1	101	alex	30	alex@gmail.com	20000	IT
2	102	bob	20	bob@gmail.com	30000	salesforce
3	103	sunny	40	sunny@gmail.com	35000	HR
4	104	stella	30	stella@gmail.com	35000	sales
5	105	nani	40	nani@gmail.com	50000	HR
6	106	stella	35	stella@gmail.com	35000	sales
7	107	nani	35	nani@gmail.com	50000	HR

Query executed successfully.

LN 110 Col 24 Ch 24 INS

DESKTOP-HPGQH4S\SQLEXPRESS ... DESKTOP-HPGQH4S\DELL (61) dataclean 00:00:00 7 rows

Ready Search 23-01-2024 14:58

Some operations using aggregations, group by and having clause

```
datacleaning & TC - DESKTOP-HPGQH4S\SQLEXPRESS.dataclean (DESKTOP-HPGQH4S\DELL (61)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Object Explorer
Connect Connect ...
DESKTOP-HPGQH4S\SQLEXPRESS (SQL Server)
Databases Security
Server Objects Replication Management XEvent Profiler
dataclean
New Query Execute
datacleaning & TC - DESKTOP-HPGQH4S\DELL (61) * X
--average salary of employees belonging to HR department
select avg(salary) as Avg_salary from employee
where department='HR';

-- Number of employees with name Stella
select count(id) as Emp_count from employee
where name='Stella';

-- Maximum salary of HR department
select max(salary)as Salary_max,department from employee
group by department
having department='HR';

--Employee count whose salary is 50000
select count(id) as Employee_count,salary from employee
where salary=50000
group by salary;
```

Results Messages

	count_employee	salary
1	2	50000

Query executed successfully.

LN 130 Col 1 Ch 1 INS

DESKTOP-HPGQH4S\SQLEXPRESS ... DESKTOP-HPGQH4S\DELL (61) dataclean 00:00:00 1 rows

Ready Search 23-01-2024 15:10

# TCL commands

## 1) Commit

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'dataclean' is selected. In the main query window, the following TCL command is run:

```
-- TCL commands(commit,rollback,savepoint)
begin transaction;
select @@TRANCOUNT -- count of transactions

delete from employee where age=20;
commit;

select * from employee;
```

The results pane displays the following table:

	id	name	age	email	salary	department
1	101	alex	30	alex@gmail.com	20000	IT
2	103	sunny	40	sunny@gmail.com	35000	HR
3	104	stella	35	stella@gmail.com	35000	sales
4	105	nani	40	nani@gmail.com	50000	HR
5	106	stella	35	stella@gmail.com	35000	sales
6	107	nani	35	nani@gmail.com	50000	HR

A message at the bottom of the results pane says "Query executed successfully."

## 2) Before using rollback after deleting data

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'dataclean' is selected. In the main query window, the following TCL command is run:

```
-- TCL commands(commit,rollback,savepoint)
begin transaction;
select @@TRANCOUNT -- count of transactions

delete from employee where age=20;
commit;

select * from employee;

delete from employee where age=30;
rollback;
```

The results pane displays the following table:

	id	name	age	email	salary	department
1	101	alex	30	alex@gmail.com	20000	IT
2	104	stella	35	stella@gmail.com	35000	sales
3	106	stella	35	stella@gmail.com	35000	sales
4	107	nani	35	nani@gmail.com	50000	HR

A message at the bottom of the results pane says "Query executed successfully."

Row with age 30 got deleted

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window displays the following T-SQL code:

```
group by salary
-- TCL commands(commit,rollback,savepoint)
begin transaction;
select @@TRANCOUNT -- count of transactions

delete from employee where age=20;
commit;

select * from employee;

delete from employee where age=30;
rollback;

select * from employee
```

The results pane shows a table with four rows of data:

	id	name	age	email	salary	department
1	104	stella	35	stella@gmail.com	35000	sales
2	106	stella	35	stella@gmail.com	35000	sales
3	107	nani	35	nani@gmail.com	50000	HR

A message at the bottom of the results pane says "Query executed successfully."

## Use rollback

The screenshot shows the Microsoft SQL Server Management Studio interface. The same T-SQL code is run, but this time the delete statement for age 30 is followed by a rollback; no commit is issued.

The results pane shows the same four rows of data as before:

	id	name	age	email	salary	department
1	101	alex	30	alex@gmail.com	20000	IT
2	104	stella	35	stella@gmail.com	35000	sales
3	106	stella	35	stella@gmail.com	35000	sales
4	107	nani	35	nani@gmail.com	50000	HR

A message at the bottom of the results pane says "Query executed successfully."

## Steps to be followed:

- 1) Run begin transaction
- 2) Run the query
- 3) Verify using select
- 4) Now run either commit or rollback
- 5) If committed values can't be modified
- 6) Using rollback we can get back the values.

## Using savepoints:

```
begin tran;  
delete from employee where id=112;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a connection to 'DESKTOP-HPGQI4S\SQLEXPRESS' is selected. In the main query window, the following T-SQL code is run:

```
-- save points  
begin tran;  
  
delete from employee where id=109;  
  
-- create save point s1  
save transaction s1  
  
rollback transaction s1  
  
select * from employee;
```

The 'Messages' pane at the bottom displays the results of the execution:

```
(1 row affected)  
Completion time: 2024-01-23T15:45:22.8171275+05:30
```

The status bar at the bottom right shows 'Query executed successfully.' and the completion time.

Rollback transcation s1

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a connection to 'DESKTOP-HPGQI4S\SQLEXPRESS' is selected. In the main query window, the same T-SQL code is run:

```
-- save points  
begin tran;  
  
delete from employee where id=109;  
  
-- create save point s1  
save transaction s1  
  
rollback transaction s1  
  
select * from employee;
```

The 'Results' pane at the bottom displays the current state of the 'employee' table:

	id	name	age	email	salary	department
1	101	sita	30	sita@gmail.com	20000	IT
2	104	stella	35	stella@gmail.com	38000	sales
3	105	stella	20	stella@gmail.com	40000	HR
4	106	stella	35	stella@gmail.com	35000	sales
5	107	nani	35	nani@gmail.com	50000	HR
6	108	nimmi	30	nimmi@gmail.com	30000	IT
7	109	kriah	30	kriah@gmail.com	45000	sales
8	110	aparna	20	aparna@gmail.com	30000	sales
9	111	kishore	35	kishore@gmail.com	35000	IT

The status bar at the bottom right shows 'Query executed successfully.' and the completion time.

## Insert command

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'datacleaning & TCL commands employee.sql' is open, displaying the following T-SQL code:

```
-- create save point s1
save transaction s1

rollback transaction s1

begin tran;
insert into employee(id,name,age,email,salary,department) values(103,'bob',45,'bob@gmail.com',30000,'IT');
--creating the savepoint s2
save transaction s2;
rollback transaction s2;

select * from employee;
```

Below the code, the results pane shows a table with 11 rows of employee data. The status bar at the bottom indicates 'Query executed successfully.'

d	name	age	email	salary	department
1	alena	30	alena@gmail.com	20000	IT
2	bob	45	bob@gmail.com	30000	IT
3	stella	35	stella@gmail.com	35000	sales
4	stella	20	stella@gmail.com	40000	HR
5	stella	35	stella@gmail.com	35000	sales
6	nani	35	nani@gmail.com	50000	HR
7	nimm	30	nimm@gmail.com	30000	IT
8	kish	30	kish@gmail.com	45000	sales
9	aparna	20	aparna@gmail.com	30000	sales

## After rollback

The screenshot shows the Microsoft SQL Server Management Studio interface. The same T-SQL code is run again, but the results show only 10 rows of data, indicating that the new row was not committed due to the rollback.

d	name	age	email	salary	department
1	alena	30	alena@gmail.com	20000	IT
2	stella	35	stella@gmail.com	35000	sales
3	stella	20	stella@gmail.com	40000	HR
4	stella	35	stella@gmail.com	35000	sales
5	nani	35	nani@gmail.com	50000	HR
6	nimm	30	nimm@gmail.com	30000	IT
7	kish	30	kish@gmail.com	45000	sales
8	aparna	20	aparna@gmail.com	30000	sales
9	kishore	35	kishore@gmail.com	35000	IT

Create another table emp and add records into it

```

datacleaning & TSQL commands employee.sql - DESKTOP-HPGQH45\SQLEXPRESS.dataclean (DESKTOP-HPGQH45\DELL (61)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query Execute
datacleaning
Object Explorer
Connect
DESKTOP-HPGQH45\SQLEXPRESS (SQL Server)
Databases Security Server Objects Replication Management XEvent Profiler
dataclean
datacleaning & TSQL commands employee.sql
-- create table emp and insert records into it
create table emp(
id int,
name varchar(10),
age int,
email varchar(20),
salary int,
department varchar(10)
)
select * from employee;
insert into emp(id,name,age,email,salary,department) values(201,'bob',45,'bob@gmail.com',30000, 'IT');
insert into emp(id,name,age,email,salary,department) values(107,'lucky',28,'lucky@gmail.com',60000, 'salesforce');
insert into emp(id,name,age,email,salary,department) values(108,'nimmi',30,'nimmi@gmail.com',30000, 'IT');
insert into emp(id,name,age,email,salary,department) values(202,'heshi',45,'heshi@gmail.com',30000, 'HR');
insert into emp(id,name,age,email,salary,department) values(203,'minnu',45,'minnu@gmail.com',30000, 'HR');

136 % 4
Messages
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)

136 %
Query executed successfully.
LN 183 Col 24 Ch 24 INS
DESKTOP-HPGQH45\SQLEXPRESS .. DESKTOP-HPGQH45\DELL (61) dataclean 00:00:00 0 rows
Ready
Search
16:19 23-01-2024

```

Retrieve records from emp table

```

datacleaning & TSQL commands employee.sql - DESKTOP-HPGQH45\SQLEXPRESS.dataclean (DESKTOP-HPGQH45\DELL (61)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query Execute
datacleaning
Object Explorer
Connect
DESKTOP-HPGQH45\SQLEXPRESS (SQL Server)
Databases Security Server Objects Replication Management XEvent Profiler
dataclean
datacleaning & TSQL commands employee.sql
select * from emp;
insert into emp(id,name,age,email,salary,department) values(201,'bob',45,'bob@gmail.com',30000, 'IT');
insert into emp(id,name,age,email,salary,department) values(107,'lucky',28,'lucky@gmail.com',60000, 'salesforce');
insert into emp(id,name,age,email,salary,department) values(108,'nimmi',30,'nimmi@gmail.com',30000, 'IT');
insert into emp(id,name,age,email,salary,department) values(202,'heshi',45,'heshi@gmail.com',30000, 'HR');
insert into emp(id,name,age,email,salary,department) values(203,'minnu',45,'minnu@gmail.com',30000, 'HR');

--retrieve records
select * from emp;

136 % 4
Results Messages
1 | 201 | bob | 45 | bob@gmail.com | 30000 | IT
2 | 107 | lucky | 28 | lucky@gmail.com | 60000 | salesforce
3 | 108 | nimmi | 30 | nimmi@gmail.com | 30000 | IT
4 | 202 | heshi | 45 | heshi@gmail.com | 30000 | HR
5 | 203 | minnu | 45 | minnu@gmail.com | 30000 | HR

136 %
Query executed successfully.
LN 186 Col 1 Ch 1 INS
DESKTOP-HPGQH45\SQLEXPRESS .. DESKTOP-HPGQH45\DELL (61) dataclean 00:00:00 5 rows
Ready
Search
16:20 23-01-2024

```

## I) UNION

```

datacleaning & TSQL commands employee.sql - DESKTOP-HPGQH45\SQLEXPRESS.dataclean (DESKTOP-HPGQH45\DELL (61)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query Execute
datacleaning
Object Explorer
Connect
DESKTOP-HPGQH45\SQLEXPRESS (SQL Server)
Databases Security Server Objects Replication Management XEvent Profiler
dataclean
datacleaning & TSQL commands employee.sql
select * from emp;
-- union
select * from employee
union
select * from emp;

136 %
Results Messages
1 | 201 | bob | 45 | bob@gmail.com | 30000 | IT
2 | 107 | lucky | 28 | lucky@gmail.com | 60000 | salesforce
3 | 108 | nimmi | 30 | nimmi@gmail.com | 30000 | IT
4 | 202 | heshi | 45 | heshi@gmail.com | 30000 | HR
5 | 203 | minnu | 45 | minnu@gmail.com | 30000 | HR
6 | 101 | alix | 30 | alix@gmail.com | 20000 | IT
7 | 102 | sara | 30 | sara@gmail.com | 30000 | sales
8 | 103 | stella | 35 | stella@gmail.com | 30000 | sales
9 | 104 | nora | 35 | nora@gmail.com | 30000 | sales
10 | 105 | rami | 35 | rami@gmail.com | 30000 | HR
11 | 106 | nimm | 30 | nimm@gmail.com | 30000 | HR
12 | 109 | lucky | 28 | lucky@gmail.com | 60000 | salesforce
13 | 110 | heshi | 45 | heshi@gmail.com | 30000 | HR
14 | 101 | minnu | 45 | minnu@gmail.com | 30000 | HR

136 %
Query executed successfully.
LN 191 Col 19 Ch 19 INS
DESKTOP-HPGQH45\SQLEXPRESS .. DESKTOP-HPGQH45\DELL (61) dataclean 00:00:00 14 rows
Ready
Search
16:22 23-01-2024

```

## 2) UNION ALL

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'datacleaning & TSQL commands employee.sql' is open, displaying the following T-SQL code:

```
-- union all
select * from employee
union all
select * from emp;
```

The results pane shows a table with 15 rows, combining data from both the 'employee' and 'emp' tables. The columns are: id, name, age, email, salary, and department.

	id	name	age	email	salary	department
1	101	alex	30	alex@gmail.com	30000	IT
2	104	stella	35	stella@gmail.com	35000	sales
3	105	alex	30	alex@gmail.com	30000	HR
4	106	stella	35	stella@gmail.com	35000	sales
5	107	alex	30	alex@gmail.com	30000	IT
6	108	remm	30	remm@gmail.com	30000	IT
7	109	kenn	30	kenn@gmail.com	45000	sales
8	110	alex	30	alex@gmail.com	30000	sales
9	111	kehone	35	kehone@gmail.com	35000	IT
10	112	lucky	28	lucky@gmail.com	60000	salesforce
11	103	alex	30	alex@gmail.com	30000	HR
12	107	lucky	28	lucky@gmail.com	60000	salesforce
13	108	remm	30	remm@gmail.com	30000	IT
14	202	heidi	45	heidi@gmail.com	30000	HR
15	203	remm	45	remm@gmail.com	30000	IT

Below the results, a message states: 'Query executed successfully.'

## 3) INTERSECT

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'datacleaning & TSQL commands employee.sql' is open, displaying the following T-SQL code:

```
-- INTERSECT
select * from employee
intersect
select * from emp;
```

The results pane shows a table with 1 row, which is the common data between the 'employee' and 'emp' tables. The columns are: id, name, age, email, salary, and department.

	id	name	age	email	salary	department
1	108	remm	30	remm@gmail.com	30000	IT

Below the results, a message states: 'Query executed successfully.'

## 4) EXCEPT

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'datacleaning & TSQL commands employee.sql' is open, displaying the following T-SQL code:

```
-- EXCEPT
select id,name,age from employee
except
select id,name,age from emp;
```

The results pane shows a table with 9 rows, containing data from the 'employee' table that is not present in the 'emp' table. The columns are: id, name, age.

	id	name	age
1	101	alex	30
2	104	stella	35
3	105	alex	30
4	106	stella	35
5	107	alex	30
6	109	kenn	30
7	110	alexma	20
8	111	kehone	35
9	112	lucky	28

Below the results, a message states: 'Query executed successfully.'