Відокремлений підрозділ Національного університету
біоресурсів і природокористування України
„Бережанський агротехнічний інститут”

Кафедра інформаційних технологій
та вищої математики

Маніпулювання даними. Мова DML.

Методичні рекомендації до виконання лабораторних робіт освітньої компоненти «Організація баз даних»

(для студентів спеціальності 122-Комп’ютерні науки»)

Бережани 2021

Укладач: Качурівський В.О., канд.пед.наук, доцент кафедри інформаційних технологій та вищої математики Бережанського агротехнічного інституту.

Рекомендовано до друку вченою радою відокремленого підрозділу Національного університету біоресурсів і природокористування України „Бережанський агротехнічний інститут" (Протокол № _1_ від 27.08.2021 р.)

Методична розробка містить довідковий матеріал із маніпулювання інформацією яка зосереджена в бази даних та інструкції для проведення лабораторних робіт. Виконання лабораторних робіт проводиться у програмі WorkBench. Побудована за простою схемою.

Кожне лабораторне заняття містить основний теоретичний матеріал та ілюстровані приклади, а також наводяться необхідні навчальні завдання, завдяки чому студенти зможуть не тільки перевірити свої знання, але й застосувати їх при додатків до веб-сторінок.

Розраховано на студентів спеціальності 122 Комп'ютерні науки та може бути корисною для усіх бажаючих.

Розглянуто та схвалено на засіданні кафедри інформаційних технологій та вищої математики
Протокол № _11_ від 24.06.2021р.

Рецензенти: Мартинюк С.В., канд.фіз.-мат.наук, доц.
Струбицька І.П., канд.техн.наук, доц..

© Качурівський В.О.

Лабораторна робота №6

Тема: Внесення даних до таблиці

Мета: Навчатися здійснювати внесення даних за допомогою команди Insert into

ТЕОРЕТИЧНІ ВІДОМОСТІ

Для додавання даних в БД в MySQL використовується команда **INSERT** , яка має наступний формальний синтаксис:

INSERT [INTO] ім'я_таблиці [(список_ товпців)] VALUES (значення1, значення2, ... значенняN)

Після висловлення **INSERT INTO** в дужках можна вказати список стовпців через кому, в які треба додавати дані, і в кінці після слова **VALUES** дужках перераховують значення які додаються для стовпців

Наприклад, нехай в базі даних productsdb є наступна таблиця Products:

```
1   CREATE DATABASE productsdb;
2   USE productsdb;
3   CREATE TABLE Products
4   (
5      Id INT AUTO_INCREMENT PRIMARY KEY,
6      ProductName VARCHAR(30) NOT NULL,
7      Manufacturer VARCHAR(20) NOT NULL,
8      ProductCount INT DEFAULT 0,
9      Price DECIMAL NOT NULL
10  );
```

Додамо в цю таблицю один рядок за допомогою наступного коду:

```
1   INSERT Products(ProductName, Manufacturer, ProductCount, Price)
2   VALUES ('iPhone X', 'Apple', 5, 76000);
```

У данно випадку значення будуть передаватися стовпцями по позиції. Тобто стопцю ProductName передається рядок "iPhone X", колонки Manufacturer - рядок "Apple" і так далі.

Важливо, щоб між значеннями і типами даних стовпців було відповідність. Так, стовпець ProductName представляє тип varchar, тобто рядок. Відповідно до цього колонки ми можемо передати строкове значення в одинарних лапках. А стобец ProductCount представляє тип int, тобто ціле число, тому даному стовпці потрібно передати цілі числа, але ніяк не рядки.

Після вдалого виконання в MySQL Workbench в поле виведення повинні з'явитися зелений маркер і повідомлення "1 row (s) affected":

Необов'язково при додаванні даних вказувати значення абсолютно для всіх стовпців таблиці. Наприклад, в прикладі вище не вказано значення для стовпця Id. Але оскільки для даного стовпця визначено атрибут AUTO_INCREMENT, то його значення буде автоматично генеруватися.

Також ми можемо опускати при додаванні такі стовпці, які підтримують значення NULL або для яких вказано значення за замовчуванням, тобто для них визначені атрибути NULL або DEFAULT. Так, в таблиці Products стовпець ProductCount має значення за замовчуванням - число 0. Тому ми можемо при додаванні опустити цей стовпець, і йому буде передаватися число 0:

```
1	INSERT Products(ProductName, Manufacturer, Price)
2	VALUES ('Galaxy S9', 'Samsung', 63000);
```

За допомогою ключових слів DEFAULT і NULL можна вказати, що в якості значення буде використовувати значення за замовчуванням або NULL відповідно:

```
1	INSERT Products(ProductName, Manufacturer, Price, ProductCount)
2	VALUES ('Nokia 9', 'HDM Global', 41000, DEFAULT);
```

або

```
1	INSERT Products(ProductName, Manufacturer, Price, ProductCount)
2	VALUES ('Nokia 9', 'HDM Global', 41000, NULL);
```

Множинне додавання

Також ми можемо додати відразу кілька рядків:

```
1	INSERT Products(ProductName, Manufacturer, Price, ProductCount)
2	VALUES
3	('iPhone 8', 'Apple', 51000, 3),
4	('P20 Lite', 'Huawei', 34000, 4),
5	('Galaxy S8', 'Samsung', 46000, 2);
```

В даному випадку в таблицю будуть додані три рядки.

ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завдання 1. Внести дані до таблиці Товари

Чаї	**Маса, г**	**Ціна для кінц. Споживача, грн**
Ранок срібносяйний (ранковий чай) * (меліса, м'ята, аніс, фенхель)	30 г	60.00
Вечір золотий (вечірній чай)* (меліса, маренка запашна, м'ята, ромашка, первоцвіт, волошка, троянда, мальва)	30 г	61.00

Квітка (для насолоди) (трава лимонна, липа, каркаде, первоцвіт*, вербена*, календула*, бузина*, волошки*, котовник*, соняшник*, листя смородини*, мальва*, троянда*,)	45 г	60.00
Зимові візерунки (зимовий чай)* (бузина, липа, первоцвіт мальва, базилік, майоран, бруньки ялини +модрини, чебрець, звіробій, шавлія, мати й мачуха)	40 г	55.00
Гоп Швіц (гіркий чай, корисний для печінки)*(кропива, деревій, цинторія, полин, звіробій, ісоп, календула)	20 г	32.00
Світанковий промінь(ранковий чай) * (насіння кропу, базилік, базилік лимонний, імбир, ромашка, чебрець, звіробій)	45 г	59.00
Сонячне сяйво (потогінні властивості)*(шипшина, липа, ферментоване листя малини і ожини, первоцвіт, звіробій)	65 г	60.00
Лісове звучання (фруктовий чай) (шипшина*, каркаде, аронія , яблука, ягоди бузини*, малини*, ожини* і смородини*, суниця* , вишня*)	170 г	61.00
~~**Лебедина пісня***~~ (лимонна трава, срібна липа, розмарин, чебрець, м'ята, одна брунька троянди)	35 г	61.00
Купальська ніч*(іван-чай ферментований, троянда, ехінацея, монарда)	48 г	102.00
Спеції:		
Сіль з травами, баночка*(сіль морська 85%, базилік, кріп, цибуля, порей, петрушка, шніт, майоран, м'ята , меліса, шавлія, чебрець, любисток)	90 г	65.00
Сіль з травами, пакет *	90 г	34.00
Сіль з травами, великий пакет *	300 г	100.00
Трави салатні, баночка *(базилік, кріп, шніт, помідори, цибуля, петрушка, морква, меліса, календула, волошки, кропива, любисток,)	15 г	65.00

Трави салатні, пакет *	15 г	29.00
Трави італійські, баночка*(орегано, майоран, базилік, розмарин, чабер, чебрець, шавлія, перець)	15 г	59.00
Трави італійські, пакет *	15 г	34.00
Суміш до м'яса, банка*(порей, часник, цибуля, гірчиця, орегано, любисток, чебрець, чабер, майоран, розмарин , шавлія)	32 г	61.00
Суміш до м'яса, пакет *	32 г	35.00
Смачна Зупа	20 г	33.00

Завдання 2. На основі текстових даних до відповідних таблиць інформацію. Імена та контактні дані змінено.

21.10.2020
ЗАМОВЛЕННЯ
Прізвище, імя: Вльона
Магазин: Вбчак
Місто: Київ
Відділення Нової Пошти: 291
Телефон: 0177901611
E-mail: helenprykhnych@com
ТОВАРИ

Квітка - 1
Зимові візерунки - 2
Сонячне сяйво - 1
Трави італійські, баночка - 1
Трави італійські, пакет – 1

16.10.2020
ЗАМОВЛЕННЯ
Прізвище, імя: Елена
Місто: Одесса
Відділення Нової Пошти: 29
Телефон: 0109818861
E-mail: leninskaua@com
ТОВАРИ

Квітка - 1
Світанковий промінь - 1
Сонячне сяйво - 1
Сіль з травами, баночка - 1
Сіль з травами, великий пакет - 1
Трави італійські, баночка – 1

16.10.2020
ЗАМОВЛЕННЯ
Прізвище, імя: Вікторія
Магазин: Беру для себе
Місто: Київ
Відділення Нової Пошти: 279
Телефон: 0167711564
E-mail: viktoriya.stet@com
ТОВАРИ

Зимові візерунки - 2
Лісове звучання - 1
Сіль з травами, великий пакет - 1
Смачна Зупа – 2

15.10.2020
ЗАМОВЛЕННЯ
Прізвище, імя: Арсен
Магазин: KAVAUA
Місто: Маріуполь
Відділення Нової Пошти: 3
Телефон: 0161120376
E-mail: kavauarob@com
ТОВАРИ

Ранок срібносяйний - 4
Вечір золотий - 4
Зимові візерунки - 2
Лісове звучання – 6

15.10.2020
ЗАМОВЛЕННЯ
Прізвище, імя: Валентина

Місто: Киев
Відділення Нової Пошти: 129
Телефон: 0101614359
E-mail: valentine.0101614359@com
ТОВАРИ

Квітка - 1
Суміш до мяса, пакет – 3

14.10.2020
ЗАМОВЛЕННЯ
Прізвище, імя: Ілона
Місто: Київ
Відділення Нової Пошти: 95
Телефон: 0152014610
E-mail: ilonach321@com
ТОВАРИ

Зимові візерунки - 1
Світанковий промінь - 2
Лісове звучання – 1

Лабораторна робота №7

Тема: Імпорт даних до таблиці БД із зовнішніх файлів
Мета: Навчитися проводити експорт даних із зовнішніх носіїв формату CSV

ТЕОРЕТИЧНІ ВІДОМОСТІ

CSV (від англ. Comma-Separated Values - значення, розділені комами) - текстовий формат , призначений для представлення табличних даних. Рядок таблиці відповідає рядку тексту, яка містить одне або кілька полів, розділених комами.

Оператор LOAD DATA зчитує рядки з текстового файлу в таблицю на дуже високій швидкості. Файл можна прочитати з хоста сервера або хоста клієнта, залежно від того, чи LOCAL надано модифікатор. LOCAL також впливає на інтерпретацію даних та обробку помилок.

LOAD DATA є доповненням до SELECT ... INTO OUTFILE. Щоб записати дані з таблиці у файл, використовуйте SELECT ... INTO OUTFILE. Щоб прочитати файл назад у таблицю, використовуйте LOAD DATA. Синтаксис пропозицій FIELDS та LINES однаковий для обох висловлювань.

LOAD DATA

```
[LOW_PRIORITY | CONCURRENT] [LOCAL]
INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[PARTITION (partition_name [, partition_name] ...)]
[CHARACTER SET charset_name]
[{FIELDS | COLUMNS}
   [TERMINATED BY 'string']
   [[OPTIONALLY] ENCLOSED BY 'char']
   [ESCAPED BY 'char']
]
[LINES
   [STARTING BY 'string']
   [TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[(col_name_or_user_var
   [, col_name_or_user_var] ...)]
[SET col_name={expr | DEFAULT}
   [, col_name={expr | DEFAULT}] ...]
```

Приклад:

```
-- дані з файлу до користувачів
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\cust.csv'
INTO TABLE customer
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS;
```

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завдання. На основі даних замовлень створити CSV файли та провести їх імпорт до відповідних таблиць

11.10.2020

ЗАМОВЛЕННЯ

Прізвище, імя: Леся

Місто: Київ

Відділення Нової Пошти: 62

Телефон: +38068 354-68-11

E-mail: BK-GVM@meta.ua
ТОВАРИ

Квітка - 3
Лісове звучання – 2

29.09.2020
ЗАМОВЛЕННЯ
Прізвище, імя: Елена
Місто: Киевская область
Відділення Нової Пошти: 1
Телефон: +380957895593
E-mail: olenaliubomyrska@com
ТОВАРИ

Ранок срібносяйний - 8
Вечір золотий - 8
Квітка - 12
Зимові візерунки - 8
Гоп Швійц - 10
Сонячне сяйво - 8
Лісове звучання - 3
Лебедина пісня - 13
Сіль з травами, баночка - 5
Сіль з травами, пакет - 10
Сіль з травами, великий пакет - 2
Трави салатні, пакет - 4
Суміш до мяса, пакет - 4
Смачна Зупа – 4

ЗАМОВЛЕННЯ
Прізвище, імя: Надежда
Магазин: Надежда
Місто: Дніпро
Відділення Нової Пошти: 27 відділення
Телефон: +380179489377
E-mail: nashepel5855@com
ТОВАРИ

Мята марокканська - 1

Сіль з травами, великий пакет - 1
Трави салатні, баночка - 1
Трави салатні, пакет - 1
Трави італійські, баночка - 1
Трави італійські, пакет - 1
Суміш до мяса, пакет - 2
Смачна Зупа – 1

ЗАМОВЛЕННЯ
Прізвище, імя: Павло
Місто: Дніпро
Відділення Нової Пошти: 46
Телефон: +380172669255
E-mail: donp.soap@com
ТОВАРИ

Ранок срібносяйний - 3
Вечір золотий - 3
Квітка - 1
Зимові візерунки - 1
Світанковий промінь - 2
Гоп Швійц - 1
Сонячне сяйво - 1
Сіль з травами, пакет - 1
Суміш до мяса, пакет – 1

30.09.2020
ЗАМОВЛЕННЯ
Прізвище, імя: Олександр
Магазин: Земледар
Місто: Івано-Франківськ
Відділення Нової Пошти: 9
Телефон: 0178371000
E-mail: alex@zem.ua
ТОВАРИ

Ранок срібносяйний - 10
Вечір золотий - 10
Квітка - 10
Зимові візерунки - 10

Світанковий промінь - 10
Гоп Швійц - 10
Сонячне сяйво - 10
Лісове звучання - 10
Сіль з травами, баночка - 10
Сіль з травами, пакет - 10
Трави салатні, баночка - 10
Трави італійські, баночка - 10
Суміш до мяса, банка – 10

25.09.2020
ЗАМОВЛЕННЯ
Прізвище, імя: Микола
Магазин: ТОВ "Зем»
Місто: Івано-Франківськ
Відділення Нової Пошти: 9
Телефон: 0960600859
E-mail: Nikolay@Zemr.ua
ТОВАРИ

Вечір золотий - 10
Квітка - 20
Зимові візерунки - 10
Світанковий промінь - 10
Гоп Швійц - 10
Лісове звучання - 10
Сіль з травами, баночка - 20
Сіль з травами, пакет - 30
Сіль з травами, великий пакет - 10
Трави салатні, пакет - 10
Трави італійські, пакет - 10
Суміш до мяса, пакет – 10

ЗАМОВЛЕННЯ
Прізвище, імя: Елена
Місто: Киевская область
Відділення Нової Пошти: 1
Телефон: 0157895593
E-mail: olenaliubomyrska@com
ТОВАРИ

Вечір золотий - 10
Квітка - 21
Зимові візерунки - 10
Гоп Швійц - 10
Лісове звучання - 5
Мята марокканська - 3
Сіль з травами, баночка - 5
Сіль з травами, пакет - 15
Сіль з травами, великий пакет - 2
Трави салатні, пакет - 5
Трави італійські, пакет - 5
Суміш до мяса, пакет - 5

Лабораторна робота №8

Тема: Оновлення даних. Команда UPDATE.
Мета: Навчитися оновлювати дані таблиць БД командою **UPDATE**

ТЕОРЕТИЧНІ ВІДОМОСТІ

Команда **UPDATE** застосовується для поновлення вже наявних рядків. Вона має наступний формальний синтаксис:

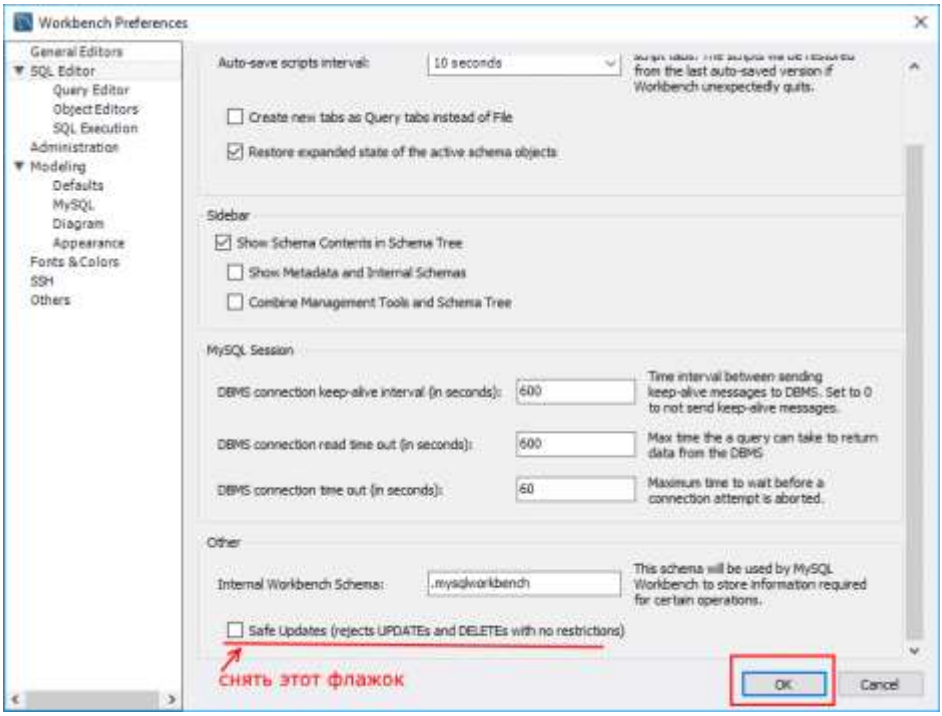
```
1   UPDATE імя_таблиці
2   SET стовпець1 = значение1, стовпець2 = значення2, ... стовпець  N
    =значенняN
        [WHERE умова оновлення]
```

Наприклад, збільшимо у всіх товарів ціну на 3000:

```
1       UPDATE Products
2       SET Price = Price + 3000;
```

Однак при виконанні даного запиту в MySQL Workbench ми можемо зіткнутися з помилкою.

Помилка говорить про те, що ми знаходимося в безпечному режимі. І щоб його відключити, в MySQL Workbench треба перейти в меню Edit -> Preferences і у вікні, перейти до пункту SQL Editor :



У вкладці, в самому низу треба зняти прапорець з поля "Safe Updates (reject UPDATEs and DELETEs with no restrictions)" і потім зберегти зміни, натиснувши на кнопку OK. Після цього треба перепідключитися до сервера.

Використовуємо вираз WHERE і змінимо назву виробника з "Samsung" на "Samsung Inc.":

```
1        UPDATE Products
2        SET Manufacturer = 'Samsung Inc.'
3        WHERE Manufacturer = 'Samsung';
```

Також можна оновлювати відразу декілька стовпців:

```
1    UPDATE Products
2    SET Manufacturer = 'Samsung',
3        ProductCount = ProductCount + 3
4    WHERE Manufacturer = 'Samsung Inc.';
```

При оновленні замість конкретних значень і виразів ми можемо використовувати ключові слова **DEFAULT** і **NULL** для установки відповідно значення за замовчуванням або NULL:

```
1    UPDATE Products
2    SET ProductCount= DEFAULT
```

3 WHERE Manufacturer = 'Huawei';

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завдання 1. Провести оновлення цін на товари згідно нового прайсу

Чаї	**Маса, г**	**Ціна для кінц. Споживача, грн**
Ранок срібносяйний (ранковий чай) * (меліса, м'ята, аніс, фенхель)	30 г	62.00
Вечір золотий (вечірній чай)* (меліса, маренка запашна, м'ята, ромашка, первоцвіт, волошка, троянда, мальва)	30 г	63.00
Квітка (для насолоди) (трава лимонна, липа, каркаде, первоцвіт*, вербена*, календула*, бузина*, волошки*, котовник*, соняшник*, листя смородини*, мальва*, троянда*,)	45 г	69.00
Зимові візерунки (зимовий чай)* (бузина, липа, первоцвіт мальва, базилік, майоран, бруньки ялини +модрини, чебрець, звіробій, шавлія, мати й мачуха)	40 г	65.00
Гоп Швіц (гіркий чай, корисний для печінки)*(кропива, деревій, цинторія, полин, звіробій, ісоп, календула)	20 г	39.00
Світанковий промінь(ранковий чай) * (насіння кропу, базилік, базилік лимонний, імбир, ромашка, чебрець, звіробій)	45 г	62.00
Сонячне сяйво (потогінні властивості)*(шипшина, липа, ферментоване листя малини і ожини, первоцвіт, звіробій)	65 г	67.00
Лісове звучання (фруктовий чай) (шипшина*, каркаде, аронія , яблука, ягоди бузини*, малини*, ожини* і смородини*, суниця* , вишня*)	170 г	69.00
~~**Лебедина пісня***~~ (лимонна трава, срібна липа, розмарин, чебрець, м'ята, одна брунька троянди)	35 г	67.00
Купальська ніч*(іван-чай ферментований, троянда, ехінацея, монарда)	48 г	110.00

Спеції:		
Сіль з травами, баночка* (сіль морська 85%, базилік, кріп, цибуля, порей, петрушка, шніт, майоран, м'ята , меліса, шавлія, чебрець, любисток)	90 г	70.00
Сіль з травами, пакет *	90 г	39.00
Сіль з травами, великий пакет *	300 г	112.00
Трави салатні, баночка *(базилік, кріп, шніт, помідори, цибуля, петрушка, морква, меліса, календула, волошки, кропива, любисток,)	15 г	70.00
Трави салатні, пакет *	15 г	39.00
Трави італійські, баночка* (орегано, майоран, базилік, розмарин, чабер, чебрець, шавлія, перець)	15 г	70.00
Трави італійські, пакет *	15 г	39.00
Суміш до м'яса, банка* (порей, часник, цибуля, гірчиця, орегано, любисток, чебрець, чабер, майоран, розмарин , шавлія)	32 г	70.00
Суміш до м'яса, пакет *	32 г	39.00
Смачна Зупа	20 г	39.00

Створити запит для виводу цін на товари для візуальної перевірки.
Завдання №2. Ввести новий чай

Купальська ніч*(іван-чай ферментований, троянда, ехінацея, монарда)	48 г	110.00

Завдання №3. У таблиці Товари Бази даних Потутори ввести позначки про відсутність товарів:
Липа, М'ята марокканська, Лебедина пісня, Flower Power.

Завдання №4. Акції на товари.
Згідно різдвяних акції вводяться знижки на таку товари:
Чаї зменшують ціну на 20%, спеції на 25%

Лабораторна робота №9 (4 год)

Тема: Вибірка інформації по таблицях Бази даних

Мета: Здійснювати вибірку даних за різноманітними умова пошуку командою SELECT

ТЕОРЕТИЧНІ ВІДОМОСТІ

Для вибірки даних з БД в MySQL застосовується команда **SELECT** . У спрощеному вигляді вона має наступний синтаксис:

SELECT список_стовпців FROM і'мя_таблиці

Наприклад, нехай раніше була створена таблиця Products, і в неї додані деякі початкові дані:

```
1   CREATE TABLE Products
2   (
3     Id INT AUTO_INCREMENT PRIMARY KEY,
4     ProductName VARCHAR(30) NOT NULL,
5     Manufacturer VARCHAR(20) NOT NULL,
6     ProductCount INT DEFAULT 0,
7     Price DECIMAL
8   );
9
10  INSERT INTO Products (ProductName, Manufacturer, ProductCount, Price)
11  VALUES
12  ('iPhone X', 'Apple', 3, 76000),
13  ('iPhone 8', 'Apple', 2, 51000),
14  ('Galaxy S9', 'Samsung', 2, 56000),
15  ('Galaxy S8', 'Samsung', 1, 41000),
16  ('P20 Pro', 'Huawei', 5, 36000);
```

Отримаємо всі об'єкти з цієї таблиці:

```
1   SELECT * FROM Products;
```

Символ зірочка * вказує, що нам треба отримати всі стовпці.

Варто зазначити, що застосування зірочки * для отримання даних вважається не дуже хорошою практикою, так як зазвичай необхідно отримати дані по невеликому набору стовпців. Тому більш оптимальний підхід полягає у вказівці всіх необхідних стовпців після слова SELECT. Виняток становить той випадок, коли треба отримати дані по абсолютно всіх стовпцях таблиці. Також використання символу * може бути переважно тоді, коли назви стовпців не відомі.

Якщо необхідно отримати дані не з усіх, а з якихось конкретних стовпців, тоді специфікації цих стовпців перераховуються через кому після SELECT:

```
1   SELECT ProductName, Price FROM Products;
```

Специфікація стовпця необов'язково повинна представляти його назву. Це може бути будь-який вираз, наприклад, результат арифметичної операції. Так, виконаємо наступний запит:

```
1  SELECT ProductName, Price * ProductCount
2  FROM Products;
```

Тут при вибірці будуть створюватися два стовпці. Причому другий стовпець представляє значення стовпця Price, помножене на значення стовпця ProductCount, тобто сукупну вартість товару.

За допомогою оператора **AS** можна змінити назву вихідного стовпця або визначити його псевдонім:

```
1  SELECT ProductName AS Title, Price * ProductCount AS TotalSum
2  FROM Products;
```

Тут для першого стовпця визначається псевдонім Title, хоча в реальності він буде представляти стовпець ProductName. Другий стовпець TotalSum зберігає твір стовпців ProductCount і Price.

Унікальні дані

За допомогою оператора **DISTINCT** можна вибрати унікальні дані за певними стовпцями.

```
1  SELECT DISTINCT Manufacturer FROM Products;
```

Сортування

Оператор **ORDER BY** сортують значення по одному або декількох стовпцях. Наприклад, впорядкуємо вибірку з таблиці Products по стовпцю Price:

```
1  SELECT * FROM Products
2  ORDER BY Price;
```

Сортування по спадаючій

За замовчуванням дані сортуються за зростанням, проте за допомогою оператора **DESC** можна задати сортування по спадаючій.

```
1  SELECT ProductName, ProductCount
2  FROM Products
3  ORDER BY ProductCount DESC;
```

За замовчуванням замість DESC використовується оператор **ASC** , який сортує по зростанню:

```
1  SELECT ProductName, ProductCount
2  FROM Products
3  ORDER BY ProductCount ASC;
```

Кількість записів

Оператор **LIMIT** дозволяє витягти певну кількість рядків і має наступний синтаксис:

```
1 LIMIT [offset,] rowcount
```

Якщо оператору LIMIT передається один параметр, то він вказує на кількість видобутих рядок. Якщо передається два параметри, то перший параметр встановлює зсув щодо початку, тобто скільки рядків потрібно пропустити, а другий параметр також вказує на кількість видобутих рядків.

Наприклад, виберемо перші три рядки:

```
1 SELECT * FROM Products
2 LIMIT 3;
```

Оператори **GROUP BY** і **HAVING** дозволяють згрупувати дані. Вони вживаються в рамках команди SELECT:

```
1 SELECT столбцы
2 FROM таблица
3 [WHERE условие_фильтрации_строк]
4 [GROUP BY столбцы_для_группировки]
5 [HAVING условие_фильтрации_групп]
6 [ORDER BY столбцы_для_сортировки]
```

GROUP BY

Оператор **GROUP BY** визначає, як рядки будуть групуватися.

Наприклад, згрупуємо товари по виробнику

```
1 SELECT Manufacturer, COUNT(*) AS ModelsCount
2 FROM Products
3 GROUP BY Manufacturer
```

Перший стовпець в вираженні SELECT - Manufacturer представляє назву групи, а другий стовпець - ModelsCount представляє результат функції Count, яка обчислює кількість рядків в групі.

Фільтрація груп. HAVING

Оператор **HAVING** дозволяє виконати фільтрацію груп, тобто визначає, які групи будуть включені в вихідний результат.

Використання HAVING багато в чому подібне до використання WHERE. Тільки є WHERE застосовується для фільтрації рядків, то HAVING - для фільтрації груп.

Наприклад, знайдемо всі групи товарів по виробникам, для яких визначено більше 1 моделі:

```
1 SELECT Manufacturer, COUNT(*) AS ModelsCount
```

```
2   FROM Products
3   GROUP BY Manufacturer
4   HAVING COUNT(*) > 1
```

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Пошукові запити здійснювати по Базі даних «ПОТУТОРИ»

Завдання на виконання

Пошукові операції по таблиці customer.

Завдання №1. Вивести інформацію про користувача з id: 3, 7, 10.

Завдання №2. Вивести інформацію про користувачів з id: 3 та 7 та 10.

Завдання №3. Вивести інформацію про користувачів з id яких менше 10.

Завдання №4. Вивести інформацію про користувачів які проживають у м.Київ, м. Дніпро

Завдання №5. Вивести інформацію про користувачів які проживають у м.Київ та м. Івано-Франківськ

Завдання №6. Вивести інформацію про користувачів яких телефон починається на 066, 096, 050, 068

Завдання №7. Вивести інформацію про користувачів які користуються телефонами мобільних операторів: life, vodafone, kyivstar

Завдання №8. Вивести інформацію про користувачів які користуються електронною поштою google.

Завдання №8-1. Вивести інформацію про користувачів які не користуються електронною поштою google.

Завдання №9. Вивести інформацію про користувачів імена яких починаються на М та П

Завдання №10. Вивести інформацію про користувачів імена яких не починаються на М та П

Завдання №11. Вивести інформацію про унікальні власників унікальних телефонів

Завдання №12. Вивести інформацію про унікальні електронні адреси.

Таблиця Товари

Завдання №13. Вивести інформацію про ціну чаю «Зимові візерунки» та його опис.

Завдання №14. Вивести назви чаїв ціна яких рівна 45 гривень.

Завдання №15. Вивести назви спецій ціна яких є від 40 до 50 гривень.

Завдання №16. Вивести інформацію про акції на (знижки) на товари.

Завдання №17. Вивести інформацію про усі спеції. Провести впорядкування назв за алфавітним порядком.(зростання)

Завдання №18. Вивести інформацію про усі чаї. Провести впорядкування назв за алфавітним порядком.(спадання)

Завдання №19. Вивести інформацію про товари від найдешевшого до найдорожчого

Завдання №20.Вивести назви товарів у яких відсутній опис.

Завдання №21. Вивести інформацію про назви товарів які містять в назві ключові слова: ранок, трави.

Таблиця list _order

Завдання №21. Вивести унікальні номера замовлень.

Завдання №22. Вивести зміст замовлення за номером ордеру за номерами: номер 5,7,12.

Завдання №22. Вивести інформацію про замовлення користувача за його ідентифікаційним кодом: 7,1,9.

Таблиця oblik

Завдання №23. Вивести інформацію про замовлення користувача з кодом №12.

Завдання №24. Вивести інформацію про замовлення користувачів у вересні, жовтні місяці.

Завдання №25. Вивести інформацію про наявність замовлень 20.09.2020, 15.09.2020.

Завдання №26. Визначити чи надавалися знижки на замовлення таким користувачами 4,12,3.

Лабораторна робота №10

Тема: Агрегатні функції та групування записів.

Мета:Навчитися застосовувати для обчислення скалярних даних

ТЕОРЕТИЧНІ ВІДОМОСТІ

Агрегатні функції обчислюють деякі скалярні значення в наборі рядків. В MySQL є наступні агрегатні функції:

- **AVG** : обчислює середнє значення
- **SUM** : обчислює суму значень
- **MIN** : обчислює найменше значення
- **MAX** : обчислює найбільше значення
- **COUNT** : обчислює кількість рядків в запиті

Все агрегатні функції приймають в якості параметра вираз, яке представляє критерій для визначення значень. Найчастіше, в якості вираження виступає назва стовпчика, над значеннями якого треба проводити обчислення.

Вирази в функціях **AVG** і **SUM** має представляти числове значення (наприклад, стовпець, який зберігає числові значення). Вираз у функціях **MIN** , **MAX** і **COUNT** може представляти числове або строкове значення або дату.

Все агрегатні функції за винятком COUNT(*)ігнорують значення NULL.

Avg

Функція **Avg** повертає середнє значення на діапазоні значень стовпця таблиці.

Наприклад, нехай є наступна таблиця товарів Products:

```
1   CREATE TABLE Products
2   (
3     Id INT AUTO_INCREMENT PRIMARY KEY,
4     ProductName VARCHAR(30) NOT NULL,
5     Manufacturer VARCHAR(20) NOT NULL,
6     ProductCount INT DEFAULT 0,
7     Price DECIMAL NOT NULL
8   );
9
10  INSERT INTO Products(ProductName, Manufacturer, ProductCount, Price)
11  VALUES
12  ('iPhone X', 'Apple', 3, 76000),
13  ('iPhone 8', 'Apple', 2, 51000),
14  ('iPhone 7', 'Apple', 5, 32000),
15  ('Galaxy S9', 'Samsung', 2, 56000),
16  ('Galaxy S8', 'Samsung', 1, 46000),
17  ('Honor 10', 'Huawei', 5, 28000),
18  ('Nokia 8', 'HMD Global', 6, 38000)
```

Знайдемо середню ціну товарів з бази даних:

```
1   SELECT AVG(Price) AS Average_Price FROM Products
```

Для пошуку середнього значення в якості вираження в функцію передається стовпець Price. Для одержуваного значення встановлюється псевдонім Average_Price, хоча в принципі встановлювати псевдонім необов'язково.

На етапі вибірки можна застосовувати фільтрацію. Наприклад, знайдемо середню ціну для товарів певного виробника:

```
1   SELECT AVG(Price) FROM Products
2   WHERE Manufacturer='Apple'
```

Також можна знаходити середнє значення для більш складних виразів.

Наприклад, знайдемо середню суму всіх товарів, з огляду на їх кількість:

```
1   SELECT AVG(Price * ProductCount) FROM Products
```

Count

Функція **Count** обчислює кількість рядків у вибірці. Є дві форми цієї функції. Перша форма COUNT(*)підраховує число рядків у вибірці:

```
1    SELECT COUNT(*) FROM Products
```

Друга форма функції обчислює кількість рядків за певним стовпцем, при цьому рядки зі значеннями NULL ігноруються:

```
1    SELECT COUNT(Manufacturer) FROM Products
```

Min і Max

Функції **Min** і **Max** обчислюють мінімальне і максимальне значення за стовпцем відповідно. Наприклад, знайдемо мінімальну ціну серед товарів:

```
1    SELECT MIN(Price), MAX(Price) FROM Products
```

Дані функції також ігнорують значення NULL і не враховують їх при підрахунку.

Sum

Функція **Sum** обчислює суму значень стовпця. Наприклад, підрахуємо загальна кількість товарів:

```
1    SELECT SUM(ProductCount) FROM Products
```

Також замість імені стовпця може передаватися обчислюється вираз. Наприклад, знайдемо загальну вартість всіх наявних товарів:

```
1    SELECT SUM(ProductCount * Price) FROM Products
```

All і Distinct

За замовчуванням всі вищеперелічених п'ять функцій враховують всі рядки вибірки для обчислення результату. Але вибірка може містити повторюють значення. Якщо необхідно виконати обчислення тільки над унікальними значеннями, виключивши з набору значень повторювані дані, то для цього застосовується оператор **DISTINCT** .

```
1    SELECT COUNT(DISTINCT Manufacturer) FROM Products
```

За замовчуванням замість DISTINCT застосовується оператор **ALL** , який вибирає всі рядки:

```
1    SELECT COUNT(ALL Manufacturer) FROM Products
```

В даному випадку ми бачимо, що виробники можуть повторюватися в таблиці, так як деякі товари можуть мати одних і тих же виробників. Тому щоб підрахувати кількість унікальних виробників, необхідно використовувати оператор DISTINCT.

Так як цей оператор неявно мається на увазі при відсутності DISTINCT, то його можна не вказувати.

Комбінування функцій

Об'єднаймо застосування декількох функцій:

```
1   SELECT COUNT(*) AS ProdCount,
2       SUM(ProductCount) AS TotalCount,
3       MIN(Price) AS MinPrice,
4       MAX(Price) AS MaxPrice,
5       AVG(Price) AS AvgPrice
6   FROM Products
```

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

COUNT()

Завдання 1. Визначити скільки заказів здійснював користувач з id=8.

Завдання 2. Визначити кількість замовлень по календарних датах в таблиці обліку замовлень.

SUM()

Завдання 3. Визначити кількість упаковок замовлених чаїв.

Завдання 4. Визначити кількість упаковок замовлених спецій.

Завдання 5. Визначити кількість замовлених упаковок чаю “Квітка”.

Завдання 6. Визначити кількість замовлених упаковок чаю “Ранок срібносяйний”.

Завдання 7. Визначити кількість замовлених упаковок чаю 'Зимові візерунки' та 'Сонячне сяйво'.

MAX()

Завдання 8. Знайти користувача який замовив найбільше чаю “Квітка”

Завдання 9. Знайти користувача який зробив найбільше замовлення.

Лабораторна робота №11

Тема: Вмонтовані функції

Мета: застосування функцій роботи із стрічками та датою для застосування у пошукових операціях

ТЕОРЕТИЧНІ ВІДОМОСТІ

Робота з рядками

Для роботи з рядками в T-SQL можна застосовувати такі функції:

- **LEN** : повертає кількість символів в рядку. Як параметр у функцію передається рядок, для якої треба знайти довжину:

```
1   SELECT LEN('Apple')  -- 5
```

- **LTRIM** : видаляє початкові пробіли з рядка. Як параметр приймає рядок:

```
1   SELECT LTRIM('  Apple')
```

- **RTRIM** : видаляє кінцеві пробіли з рядка. Як параметр приймає рядок:

```
1    SELECT RTRIM(' Apple    ')
```

- **CHARINDEX** : повертає індекс, за яким знаходиться перше входження підрядка в рядку. В якості першого параметра передається подстрока, а в якості другого - рядок, в якій треба вести пошук:

```
1    SELECT CHARINDEX('pl', 'Apple') -- 3
```

- **PATINDEX** : повертає індекс, за яким знаходиться перше входження певного шаблону в рядку:

```
1    SELECT PATINDEX('%p_e%', 'Apple')   -- 3
```

- **LEFT** : вирізає з початку рядка певну кількість символів. Перший параметр функції - рядок, а другий - кількість символів, які треба вирізати спочатку рядки:

```
1    SELECT LEFT('Apple', 3) -- App
```

- **RIGHT** : вирізає з кінця рядка певну кількість символів. Перший параметр функції - рядок, а другий - кількість символів, які треба вирізати спочатку рядки:

```
1    SELECT RIGHT('Apple', 3)    -- ple
```

- **SUBSTRING** : вирізає з рядка підрядок певною довжиною, починаючи з певного індексу. Певий параметр функції - рядок, другий - початковий індекс для вирізки, і третій параметр - кількість вирізаних символів:

```
1    SELECT SUBSTRING('Galaxy S8 Plus', 8, 2)    -- S8
```

- **REPLACE** : замінює одну подстроку інший в рамках рядки. Перший параметр функції - рядок, другий - підрядок, яку треба замінити, а третій - підрядок, на яку треба замінити:

```
1    SELECT REPLACE('Galaxy S8 Plus', 'S8 Plus', 'Note 8')   -- Galaxy Note 8
```

- **REVERSE** : перевертає рядок навпаки:

```
1    SELECT REVERSE('123456789') -- 987654321
```

- **CONCAT** : об'єднує два рядки в одну. Як параметр приймає від 2-х і більше рядків, які треба з'єднати:

```
1    SELECT CONCAT('Tom', ' ', 'Smith')  -- Tom Smith
```

- **LOWER** : переводить рядок в нижній регістр:

```
1    SELECT LOWER('Apple')   -- apple
```

- **UPPER** : переводить рядок у верхній регістр

```
1    SELECT UPPER('Apple')   -- APPLE
```

- **SPACE** : повертає рядок, яка містить певну кількість пропусків

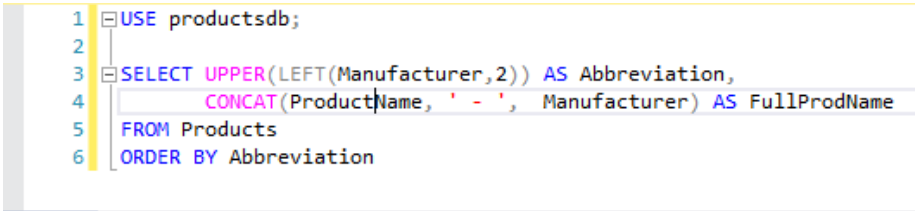
Наприклад, візьмемо таблицю:

```
1    CREATE TABLE Products
2    (
3        Id INT IDENTITY PRIMARY KEY,
4        ProductName NVARCHAR(30) NOT NULL,
5        Manufacturer NVARCHAR(20) NOT NULL,
6        ProductCount INT DEFAULT 0,
7        Price MONEY NOT NULL
```

```
8    );
```

І при отриманні даних застосуємо рядкові функції:

```
1    SELECT UPPER(LEFT(Manufacturer,2)) AS Abbreviation,
2        CONCAT(ProductName, ' - ',  Manufacturer) AS FullProdName
3    FROM Products
4    ORDER BY Abbreviation
```



	Abbreviation	FullProdName
1	AP	iPhone 6 - Apple
2	AP	iPhone 6S - Apple
3	AP	iPhone 7 - Apple
4	ON	OnePlus 5 - OnePlus
5	SA	Galaxy S8 - Samsung
6	SA	Galaxy S8 Plus - Samsung
7	XI	Mi 5X - Xiaomi

T-SQL надає ряд функцій для роботи з датами і часом:

- **GETDATE** : повертає поточну локальну дату і час на основі системного годинника у вигляді об'єкта datetime

```
1    SELECT GETDATE()    -- 2017-07-28 21:34:55.830
```

- **GETUTCDATE** : повертає поточну локальну дату і час за Гринвічем (UTC / GMT) у вигляді об'єкта datetime

```
1    SELECT GETUTCDATE()    -- 2017-07-28 18:34:55.830
```

- **SYSDATETIME** : повертає поточну локальну дату і час на основі системного годинника, але відміну від GETDATE полягає в тому, що дата і час повертаються у вигляді об'єкта datetime2

```
1    SELECT SYSDATETIME()       -- 2017-07-28 21:02:22.7446744
```

- **SYSUTCDATETIME** : повертає поточну локальну дату і час за Гринвічем (UTC / GMT) у вигляді об'єкта datetime2

```
1    SELECT SYSUTCDATETIME()    -- 2017-07-28 18:20:27.5202777
```

- **SYSDATETIMEOFFSET** : повертає об'єкт datetimeoffset (7), який містить дату і час щодо GMT

```
1    SELECT SYSDATETIMEOFFSET()     -- 2017-07-28 21:02:22.7446744 +03:00
```

- **DAY** : повертає день дати, який передається в якості параметра

```
1    SELECT DAY(GETDATE())    -- 28
```

- **MONTH** : повертає місяць дати

```
1    SELECT MONTH(GETDATE())    -- 7
```

- **YEAR** : повертає рік з дати

```
1    SELECT YEAR(GETDATE())    -- 2017
```

- **DATENAME** : повертає частину цієї дати у вигляді рядка. Параметр вибору частині дати передається в якості першого параметра, а сама дата передається в якості другого параметра:

```
1    SELECT DATENAME(month, GETDATE())    -- July
```

- Для визначення частини дати можна використовувати наступні параметри (в дужках вказані їх скорочені версії):
  - year (yy, yyyy): рік
  - quarter (qq, q): квартал
  - month (mm, m): місяць
  - dayofyear (dy, y): День року
  - day (dd, d): День місяця
  - week (wk, ww): тиждень
  - weekday (dw): день тижня
  - hour (hh): годину
  - minute (mi, n): хвилина
  - second (ss, s): секунда
  - millisecond (ms): мілісекунда
  - microsecond (mcs): мікросекунда
  - nanosecond (ns): наносекунд
  - tzoffset (tz): Змішання в хвилинах щодо Гринвічем (для об'єкта datetimeoffset)
- **DATEPART** : повертає частину цієї дати у вигляді числа. Параметр вибору частині дати передається в якості першого параметра (використовуються ті ж параметри, що і для DATENAME), а сама дата передається в якості другого параметра:

```
1    SELECT DATEPART(month, GETDATE())    -- 7
```

- **DATEADD** : повертає дату, яка є результатом складання числа до певного компоненту дати. Перший параметр представляє компонент дати, описаний вище для функції DATENAME. Другий параметр - додається кількість. Третій параметр - сама дата, до якої треба зробити додаток:

```
1    SELECT DATEADD(month, 2, '2017-7-28')    -- 2017-09-28 00:00:00.000
2    SELECT DATEADD(day, 5, '2017-7-28')    -- 2017-08-02 00:00:00.000
3    SELECT DATEADD(day, -5, '2017-7-28')    -- 2017-07-23 00:00:00.000
```

- Якщо додається кількість представляє негативне число, то фактично відбувається зменшення дати.
- **DATEDIFF** : повертає різницю між двома датами. Перший параметр - компонент дати, який вказує, в яких одиницях варто вимірювати різницю. Другий і третій параметри - порівнювані дати:
  SELECT DATEDIFF(year, '2017-7-28', '2018-9-28') -- різниця 1 рік
  SELECT DATEDIFF(month, '2017-7-28', '2018-9-28') -- різниця 14 місяців
  SELECT DATEDIFF(day, '2017-7-28', '2018-9-28') -- різниця 427 днів
- **TODATETIMEOFFSET** : повертає значення datetimeoffset, яке є результатом складання тимчасового зсуву з іншим об'єктом datetimeoffset

```
1  SELECT TODATETIMEOFFSET('2017-7-28 01:10:22', '+03:00')
```

- **SWITCHOFFSET** : повертає значення datetimeoffset, яке є результатом складання тимчасового зсуву з об'єктом datetime2

```
1  SELECT SWITCHOFFSET(SYSDATETIMEOFFSET(), '+02:30')
```

- **EOMONTH** : повертає дату останнього дня для місяця, який використовується в переданої в якості параметра датою.

```
1  SELECT EOMONTH('2017-02-05')  -- 2017-02-28
2  SELECT EOMONTH('2017-02-05', 3) -- 2017-05-31
```

- Як необов'язкового другого параметра можна передавати кількість місяців, які необхідно додати до дати. Тоді останній день місяця буде обчислюватися для нової дати.
- **DATEFROMPARTS** : за роком, місяцем і дня створює дату

```
1  SELECT DATEFROMPARTS(2017, 7, 28)  -- 2017-07-28
```

- **ISDATE** : перевіряє, чи є вираз датою. Якщо є, то повертає 1, інакше повертає 0.

```
1  SELECT ISDATE('2017-07-28')  -- 1
2  SELECT ISDATE('2017-28-07')  -- 0
3  SELECT ISDATE('28-07-2017')  -- 0
4  SELECT ISDATE('SQL')  -- 0
```

Як приклад використання функцій можна привести створення таблиці замовлень, яка містить дату замовлення:

```
1  CREATE TABLE Orders
2  (
3     Id INT IDENTITY PRIMARY KEY,
4     ProductId INT NOT NULL,
5     CustomerId INT NOT NULL,
6     CreatedAt DATE NOT NULL DEFAULT GETDATE(),
7     ProductCount INT DEFAULT 1,
8     Price MONEY NOT NULL
9  );
```

Вираз DEFAULT GETDATE() вказує, що якщо при додаванні даних не передається дата, то вона автоматично обчислюється за допомогою функції GETDATE ().

Інший приклад - знайдемо замовлення, які були зроблені 16 днів тому:

```
SELECT * FROM Orders
WHERE DATEDIFF(day, CreatedAt, GETDATE()) = 16
```

```
USE productsdb;

SELECT * FROM Orders
WHERE DATEDIFF(day, CreatedAt, GETDATE()) = 16
```

100 %

Results | Messages

	Id	ProductId	CustomerId	CreatedAt	ProductCount	Price
1	2	2	1	2017-07-13	1	41000,00

Функції CASE і IIF

Функція **CASE** перевіряє значення деякого вираження, і в залежності від результату перевірки може повертати той чи інший результат.

CASE приймає наступну форму:

```
CASE выражение
   WHEN значення_1 THEN результат_1
   WHEN значення_2 THEN результат_2
   .....
   WHEN значеннння_N THEN результат_N
   [ELSE альтернативний_результат]
END
```

Візьмемо для прикладу таку таблицю Products:

```
CREATE TABLE Products
(
   Id INT IDENTITY PRIMARY KEY,
   ProductName NVARCHAR(30) NOT NULL,
   Manufacturer NVARCHAR(20) NOT NULL,
   ProductCount INT DEFAULT 0,
   Price MONEY NOT NULL
);
```

Виконаємо запит до цієї таблиці і використовуємо функцію CASE:

```
1    SELECT ProductName, Manufacturer,
2      CASE ProductCount
3        WHEN 1 THEN 'Товар закінчується'
4        WHEN 2 THEN 'Мало товару'
5        WHEN 3 THEN 'Є в наявності'
6        ELSE 'Багато товару'
7      END AS EvaluateCount
8    FROM Products
```

Тут значення стовпця ProductCount послідовно порівнюється зі значеннями після операторів WHEN. Залежно від значення стовпця ProductCount функція CASE буде повертати одну з рядків, яка йде після відповідного оператора THEN. Для повертається результату визначено стовпець EvaluateCount:

Також функція **CASE** може приймати ще одну форму:

```
1    CASE
2      WHEN вираз_1 THEN результат_1
3      WHEN вираз_2 THEN результат_2
4      .....................................
5      WHEN вираз_N THEN результат_N
6      [ELSE альтернативний_результат]
7    END
```

Наприклад, стосовно таблиці Products:

```
1    SELECT ProductName, Manufacturer,
2      CASE
3        WHEN Price > 50000 THEN 'Категорія A'
4        WHEN Price BETWEEN 40000 AND 50000 THEN 'Категорія B'
5        WHEN Price BETWEEN 30000 AND 40000 THEN 'Категорія C'
6        ELSE 'Категорія D'
7      END AS Category
8    FROM Products
```

Фактично все те ж саме, що і в попередньому прикладі, тільки після CASE не вказується порівняльне значення. А самі вирази порівняння стоять після оператора WHEN. І якщо вираз після оператора WHEN було це слово, то повертається значення, яке йде після відповідного оператора THEN.

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завдання 1. Створити запит для виведення інформації замовлення за жовтень місяць.

Завдання 2. Створити запит для виведення інформації замовлення за жовтень та листопад місяць.

Завдання 3. Створити запит для виведення інформації замовлення за останніх 10 днів.

Завдання 4. Вивести назви чаїв великими літерами.

Завдання 5. Вивести назви замовлення з певним ID таким чином Чай КВІТКА, спеція СУМІШ ДО М’ЯСА.

Завдання 6. Визначити категорії замовлень: при сумі більшій за 2000 – Категорія А, інші Категорія Б.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. METANIT.COM. Сайт о программировании. Руководство по MySQL: [Електронний ресурс] – Режими доступу: https://metanit.com/sql/mysql/

2. SQL Учебник. schoolsw3.com: [Електронний ресурс] – Режим доступу: https://schoolsw3.com/sql/index.php

3. Руководства по SQL Server: . [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/sql/sql-server/tutorials-for-sql-server-2016?view=sql-server-ver15

4. SQL Підручник. W3schoolsUA. Українською. [Електронний ресурс] – Режим доступу: https://w3schoolsua.github.io/sql/index.html

Методичне видання

Качурівський В.О.

Маніпулювання даними. Мова DML.

Методичні рекомендації до виконання лабораторних робіт освітньої компоненти «Організація баз даних»

(для студентів спеціальності 122-Комп'ютерні науки»)

Підписано до друку __.__.20__р. Вид № __
Формат 60х84 1/16 Папір офсетний.
Друк на різографі. Гарнітура Times/
Умовно-друк. арк. __ Облік-вид. арк. ____
Тираж 100 прим. Замовлення № ___.

ВІКТ БАТІ
м. Бережани