Structured Query Language

CSci 150 – Fundamentals of Database Systems

Rodney M. Maniego Jr.
Instructor I / DCST / VSU

Version: 20231110

COURSE OUTCOME

Apply SQL in automating business processes.

LEARNING OUTCOME

- Identify and explain the concepts about Structured Query Language.
- Apply and demonstrate database definition, table manipulation, basic data retrieval and summarization.

TOPICS

1. Structured Query Language
2. Naming Conventions
3. DBMS Data Types
4. Data Definition Language
5. Table Manipulation
6. Basic Data Retrieval
7. Data Summarization

Structured Query Language

Structured Query Language

- A specialized programming language designed for querying data, manipulating records, and managing the structure of a relational database.
  - Originated from the “Structured English Query Language (SEQUEL)” IBM research project in the 1970s.
  - The initial work on standardizing SQL started in the 1980s.

Major Components of SQL

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)

Data Definition Language

- It used for create and modify the database.
- Common DDL commands:
  - `CREATE`: creates a new database object, such as a table or index.
  - `ALTER`: modifies the structure of an existing database object.
  - `DROP`: deletes a database object, such as a table or index.
  - `TRUNCATE`: removes all records from a table but retains its structure.

Data Manipulation Language

- It used for manipulating data stored in the database.
- Common DML commands:
  - `SELECT`: retrieves data from one or more tables.
  - `INSERT`: adds new records to a table.
  - `UPDATE`: modifies existing records in a table.
  - `DELETE`: removes records from a table.

Data Control Language

- It used for control and management of access to the database.
- Common DDL commands:
  - `GRANT`: provides specific privileges to database users.
  - `REVOKE`: removes specific privileges from database users.

Naming Conventions

Naming Conventions

- The naming conventions varies greatly.
- The basic rule is **be consistent**.

Consistency helps maintain database readability, maintainability, and organization.

Naming Databases

- General Guidelines:
  - **Consistency.** What is the best practices of your organization?
  - **Flexibility.** How flexible is the proposed database name?
  - **Scope.** What is the scope of the database?

- Use concise, descriptive, and meaningful names.
- Can be purpose-specific or project-specific DB name.
- `name_database` or `NameDB` is not required.

EXAMPLE #1: `library_queuing_system`
EXAMPLE #2: `FarmersCooperative`
EXAMPLE #3: `lolaSariSariStore`

Naming Tables

- ❑ Use singular nouns for table names.
- ❑ Use concise, descriptive, and meaningful names.
- ❑ Be consistent with the overall naming convention.
- ❑ `name_data` or `NameTable` is not required.

EXAMPLE #1: `product_category`
EXAMPLE #2: `CustomerInformation`
EXAMPLE #3: `orderItem`

Naming Columns

- ❑ Use concise, descriptive, and meaningful names.
- ❑ Use `column_id` or `ColumnID` on primary keys.
- ❑ Be consistent with the overall naming convention.

EXAMPLE #1: `user_id`
EXAMPLE #2: `InventoryID`
EXAMPLE #3: `salesID`

Naming Views

- ❑ Use `column_view` or `ColumnView` as suffix.

EXAMPLE #1: `sales_view`
EXAMPLE #2: `TransactionView`
EXAMPLE #3: `userView`

DBMS Data Types

What Data Type to Use?

- ❑ It should directly mirror the nature of the data being stored.
- ❑ Each column should align with the real-world domain of the attribute it represents.

Aligning data types with the real-world domain ensures data accuracy, integrity, and enhances the efficiency of querying and processing data within a database.

DBMS Support

- https://docs.**oracle**.com/en/database/oracle/oracle-database/19/sqlrf/Data-Types.html
- https://www.**postgresql**.org/docs/current/datatype.html
- https://learn.**microsoft**.com/en-us/**sql**/t-sql/data-types/data-types-transact-sql
- https://**mariadb**.com/kb/en/data-types/
- https://www.**sqlite**.org/datatype3.html

NOTE: Always refer to the documentation of the specific database you are working with to understand its supported data types and syntax.

INT (Integer)

- ❑ Exact-number data types that use integer data.
- ❑ Used for identifiers, counts, or quantities of whole numbers.
- ❑ Signed integer from -2,147,483,648 to 2,147,483,647.

Use the smallest `INT` data type that can reliably contain all possible values to save up database space.

EXAMPLE #1: `order_id INT`
EXAMPLE #2: `total_box_received INT`

Data type	Range	Range expression	Storage
bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	-2^63 to 2^63-1	8 Bytes
int	-2,147,483,648 to 2,147,483,647	-2^31 to 2^31-1	4 Bytes
smallint	-32,768 to 32,767	-2^15 to 2^15-1	2 Bytes
tinyint	0 to 255	2^0-1 to 2^8-1	1 Byte

Synonyms

`INT1` – `TINYINT`
`INT2` – `SMALLINT`
`INT3` – `MEDIUMINT`
`INT4` – `INT`
`INT8` – `BIGINT`

https://learn.microsoft.com/en-us/sql/t-sql/data-types/int-bigint-smallint-and-tinyint-transact-sql?view=sql-server-ver16

BOOLEAN

- Used for true or false values.
- Not a standard data type for SQL.
- Synonyms: `BIT(1)` / `NUMBER(1)` / `TINYINT(1)`.
- Supported aliases for values: `0` – `FALSE`, `1` – `TRUE`.

Usage varies across various systems,

```
EXAMPLE #1: is_canceled BOOLEAN
EXAMPLE #2: is_archived BOOLEAN
EXAMPLE #3: is_deleted BOOLEAN
```

FLOAT (Single Precision)

- Approximate-number used with floating-point numeric data.
- Used for numeric values with decimals such as measurements, monetary amounts, or percentages.
- Synonym for `FLOAT(24)` is `REAL`.
- Synonym for `FLOAT(53)` is `DOUBLE PRECISION`.

```
SYNTAX: column_name FLOAT(1-53*)

EXAMPLE #1: distance_in_km FLOAT
EXAMPLE #2: serving_size FLOAT(2)
EXAMPLE #3: total_sales DOUBLE PRECISION
EXAMPLE #4: temperature_in_celsius REAL
```

NUMERIC (Arbitrary Precision Numbers)

- Can store numbers with a very large number of digits.
- Recommended for storing monetary amounts and other quantities where exactness is required.
- Synonym for `NUMERIC(p,s)` is `DECIMAL(p,s)`.

Precision (p, 1-1000) is the total number of digits and scale (s, 0<p) is the maximum mantissa, arithmetic operations may cause errors on overflows.

```
SYNTAX: column_name NUMERIC(1000,1000)

EXAMPLE #1: bitcoin_mined NUMERIC(20,8)
EXAMPLE #2: ethereum_staked NUMERIC(38,18)
EXAMPLE #3: annual_interesdt_rate NUMERIC(5,4)
EXAMPLE #4: planck_constant NUMERIC(20,15)
```

VARCHAR (Variable Character)

- Fixed-size, **blank-padded** string data.
- Supports UTF-8 encoding, review DBCS/Multi-language strings.
- Synonym is `CHARACTER(n)`, `n`=bytes.

SYNTAX: `column_name CHARACTER(1-255-8000)`

EXAMPLE #1: `course_code CHARACTER(8)`
EXAMPLE #2: `game_account_code CHARACTER(25)`

CHAR (Fixed-Length Character)

- ❑ Fixed-size, **blank-padded** string data.
- ❑ Supports UTF-8 encoding, review DBCS/multi-language strings.
- ❑ Synonym is `CHARACTER(n)`, `n`=bytes.

SYNTAX: `column_name CHAR(1-MAX)`

EXAMPLE #1: `course_code CHAR(8)`
EXAMPLE #2: `game_account_code CHAR(25)`

Maximum Size Support

- MySQL/MariaDB – `255`
- PostgreSQL – `1073741823`
- Microsoft SQL Server – `8000`
- MySQL/MariaDB – `2000`

VARCHAR (Variable-Length Character)

- ❑ Used for textual data of variable length.
- ❑ Supports UTF-8 encoding, review DBCS/multi-language strings.
- ❑ Synonym is `CHARACTER VARYING(n)`, `n`=bytes.

SYNTAX: `column_name VARCHAR(1-MAX)`

EXAMPLE #1: `first_name VARCHAR(512)`
EXAMPLE #2: `movie_description VARCHAR(1024)`

Maximum Size Support

- MySQL/MariaDB – `65535`
- PostgreSQL – `1073741823`
- Microsoft SQL Server – `8000`
- MySQL/MariaDB – `4000`

TEXT

- Used for large textual data of variable length.
- Supports UTF-8 encodings, review DBCS/multi-language strings.
- Maximum size varies from 64B to 4GB in various DBMSs.

SYNTAX: `column_name TEXT`

EXAMPLE #1: `blog_post TEXT`
EXAMPLE #2: `document_data TEXT`

BLOB (Binary Large Object)

- Used for large textual data of variable length.
- Supports UTF-8 encodings, review DBCS/multi-language strings.
- Maximum size varies from 2GB to 4GB in various DBMSs.

SYNTAX: `column_name BLOB`

EXAMPLE #1: `thumbmark_data BLOB`
EXAMPLE #2: `aerial_image_jpg BLOB`

DATE/TIME

- Used for storing supported date/time string.
- Input varies in different DBMSs.

```
SYNTAX: column_name DATE
SYNTAX: column_name TIME
SYNTAX: column_name DATETIME
SYNTAX: column_name TIMESTAMP
```

```
EXAMPLE #1: birthdate DATE
EXAMPLE #2: schedule_start TIME
EXAMPLE #3: date_added DATETIME
EXAMPLE #4: date_modified TIMESTAMP
```

SQLite Type Affinity

- Data type is associated with the value, not with its container.
- Uses a more general dynamic type system.

MOST DBMS	Resulting Affinity	Affinity Determination
INT, INTEGER. TINYINT, SMALLINT, MEDIUMINT, BIGINT, UNSIGNED BIG INT, INT2, INT8	INTEGER	1
CHARACTER(20), VARCHAR(255), VARYING CHARACTER(255), NCHAR(55), NATIVE CHARACTER(70), NVARCHAR(100), TEXT, CLOB	TEXT	2
BLOB	BLOB	3
FLOAT, REAL, DOUBLE, DOUBLE PRECISION	REAL	4
NUMERIC, DECIMAL(10,5), BOOLEAN, DATE, DATETIME	NUMERIC	5

NOTE: Dates and Times can be stored as ISO 8601 ("YYYY-MM-DD HH:MM:SS.SSS") strings, Julian Day Real Numbers, or Unix Time Integers.

Data Definition Language

DDL / CREATE DATABASE

- It is a statement used to create a database.

SYNTAX #1: `CREATE DATABASE database_name;`
SYNTAX #2: `CREATE DATABASE IF NOT EXISTS database_name;`

DB BROWSER

```
1 CREATE DATABASE IF NOT EXISTS ecommerce_system;
2
3
4
5
```

Creates a new `ecommerce_system` database if it is not yet existing.

DDL / DROP DATABASE

- It is a statement used to delete a database.

SYNTAX #1: `DROP DATABASE database_name;`

DB BROWSER

```
1 DROP DATABASE ecommerce_system;
2
3
4
5
```

Deletes the `ecommerce_system` database, if not existing, no operation will happen.

DDL / CREATE TABLE

- It is a statement used to create a new.

SYNTAX #1: `CREATE TABLE IF NOT EXISTS table_name (...);`

DB BROWSER

```sql
CREATE TABLE IF NOT EXISTS product (
    product_id INT PRIMARY KEY AUTO_INCREMENT,
    product_name VARCHAR(256),
    prod_cat_id INT,
    FOREIGN KEY (prod_cat_id) REFERENCES product_category (prod_cat_id);
);
```

Creates a new `product` table, if it is not yet created.

DDL / ALTER TABLE

- It is a statement used to modify schema or structure of the table, such as name or its column.

SYNTAX #1: `ALTER TABLE table_name [ACTION];`

DB BROWSER

```
--- ALTER TABLE product TO product_data;
ALTER TABLE product
RENAME TO product_data;
```

Changes the `product` table name to `product_data`.

DDL / ALTER TABLE

- It is a statement used to modify schema or structure of the table, such as name or its column.

SYNTAX #1: `ALTER TABLE table_name [ACTION];`

DB BROWSER

```
ALTER TABLE product_data
ADD COLUMN product_thumbnail BLOB,
ADD COLUMN product_brand_name VARCHAR(256),
ADD COLUMN product_net_weight NUMERIC(5,2) DEFAULT NULL
ADD COLUMN product_added DATETIME DEFAULT CURRENT_TIMESTAMP;
```

Add new columns to the `product_data` table.

DDL / ALTER TABLE

- It is a statement used to modify schema or structure of the table, such as name or its column.

SYNTAX #1: `ALTER TABLE table_name [ACTION];`

DB BROWSER

```
1 ALTER TABLE product_data
2 ADD COLUMN product_data_id INT PRIMARY KEY AUTO_INCREMENT;
3
4
5
6
```

Add a primary key to the `product_data` table.

DDL / ALTER TABLE

- It is a statement used to modify schema or structure of the table, such as name or its column.

SYNTAX #1: `ALTER TABLE table_name [ACTION];`

DB BROWSER

```
ALTER TABLE product_data
ADD COLUMN manufacturer_id INT
ADD CONSTRAINT fk_manufacturer
FOREIGN KEY (manufacturer_id) REFERENCES manufacturer (manufacturer_id);
```

Add a new foreign key in the `product_data` table.

DDL / ALTER TABLE

- It is a statement used to modify schema or structure of the table, such as name or its column.

SYNTAX #1: `ALTER TABLE table_name [ACTION];`

DB BROWSER

```
1 ALTER TABLE product_data
2 CHANGE COLUMN product_brand_name product_name VARCHAR(256);
3
4
5
6
```

Rename a column name in the `product_data` table.

DDL / ALTER TABLE

- It is a statement used to modify schema or structure of the table, such as name or its column.

SYNTAX #1: `ALTER TABLE table_name [ACTION];`

DB BROWSER

```
1 ALTER TABLE product_data
2 DROP COLUMN product_name;
3
4
5
6
```

Delete a column in the `product_data` table.

DDL / TRUNCATE TABLE

- It is a statement used to *clear* all the contents of a table.

SYNTAX #1: `TRUNCATE TABLE table_name;`

DB BROWSER

```
1 TRUNCATE TABLE product_data;
2
3
4
5
6
```

Delete the contents of the `product_data` table, but not the table itself.

DDL / DROP TABLE

- It is a statement used to delete a table including its contents.

SYNTAX #1: `DROP TABLE table_name;`

DB BROWSER

```
1 DROP TABLE product_data;
2
3
4
5
6
```

Delete the `product_data` table and its contents.

Data Manipulation Language

Comparison Operators

- = equal to
- <> not equal to
- < less than
- <= less than or equal
- > greater than
- >= greater than or equal to.

Logical Operators

- **AND** and (Conjunction)
- **OR** or (Disjunction)
- **NOT** not (Negation)

Other WHERE Operations

- column IN (value1, value2, …, value3)
- BETWEEN value1 AND value2
- column LIKE “pattern%”
- column LIKE “%pattern”
- column LIKE “%pattern%”
- column IS NULL

Manipulating Table Data

- How can a table be manipulated?
  - Inserting new rows.
  - Updating specific rows and columns.
  - Deleting specific rows from a table.

Manipulating Table Data

- It used for manipulating data stored in the database.
- Three basic statements used:
  - `INSERT`: adds new records to a table.
  - `UPDATE`: modifies existing records in a table.
  - `DELETE`: removes records from a table.

DML / INSERT INTO

- It is a statement used to add new record/s to a table.

SYNTAX:

```
INSERT INTO table_name
(column1, column2, ..., columnN)
VALUES
(value1, value1, ..., valueN),
(value1, value1, ..., valueN),
...
(value1, value1, ..., valueN);
```

DML / INSERT INTO

- It is a statement used to add new record/s to a table.

SYNTAX: `INSERT INTO table_name (...) VALUES (...);`

DB BROWSER

```
INSERT INTO order_item
(order_id, inventory_id, price_per_unit, order_qty, sub_total)
VALUES
(211, 591, 149.99, 5, 749.95),
(211, 107, 998.00, 1, 998.00),
(211, 423, 572.21, 2, 1144.42);
```

Adding three products to the `order_item` table.

NOTE: Since the primary keys set to be unique and auto-increments, there is no need to manually set their values upon insertion.

DML / INSERT INTO

Account

AID	FirstName	LastName	Username	BirthDate	Role

AID	FirstName	LastName	Username	BirthDate	Role
1	Juan	Dela Cruz	OneDy	2000-09-11	0
2	Maria	Sta. Ana	lilLamb	1999-06-29	0

DML / INSERT INTO

Account

AID	FirstName	LastName	Username	BirthDate	Role

AID	FirstName	LastName	Username	BirthDate	Role
1	Juan	Dela Cruz	OneDy	2000-09-11	0
2	Maria	Sta. Ana	lilLamb	1999-06-29	0

```
INSERT INTO Account
(FirstName, LastName, Username, BirthDate, Role)
VALUES
(“Juan” , “Dela Cruz” , “OneDy” , “2000-09-11” , 0),
(“Maria” , “Sta. Ana” , “lilLamb” , “1999-06-29” , 0);
```

DML / UPDATE

- It is a statement used to update specific rows in a table.

SYNTAX:

```
UPDATE table_name
SET column1=value1,
    column2=value2,
    ...
    columnN=valueN,
WHERE condition1 AND (NOT condition2 OR condition3);
```

NOTE: Always include the WHERE clause, otherwise multiple data can be updated accidentally.

DML / UPDATE

- It is a statement used to update specific rows in a table.

SYNTAX: `UPDATE table_name SET ... WHERE ...;`

DB BROWSER

```
UPDATE order_item
SET inventory_id=89, price_per_unit=53.11, order_qty=3, sub_total=159.33
WHERE order_item_id=3;
```

Updating the order information in the `order_item` table.

NOTE: Since the primary keys set to be unique and auto-increments, there is no need to manually set their values upon insertion.

DML / UPDATE

Account

AID	FirstName	LastName	Username	BirthDate	Role
1	Juan	Dela Cruz	OneDy	2000-09-11	0
2	Maria	Sta. Ana	lilLamb	1999-06-29	0

AID	FirstName	LastName	Username	BirthDate	Role
1	Juan	Dela Cruz	OneDy	2000-09-11	0
2	Maria	Sta. Ana	admin01	1999-06-29	1

DML / UPDATE

Account

AID	FirstName	LastName	Username	BirthDate	Role
1	Juan	Dela Cruz	OneDy	2000-09-11	0
2	Maria	Sta. Ana	lilLamb	1999-06-29	0

AID	FirstName	LastName	Username	BirthDate	Role
1	Juan	Dela Cruz	OneDy	2000-09-11	0
2	Maria	Sta. Ana	admin01	1999-06-29	1

```
UPDATE Account
SET Username=“admin01”, Role=1
WHERE AID=2;
```

DML / DELETE

- It is a statement used to update specific rows in a table.

SYNTAX:
```
DELETE FROM table_name
WHERE condition1 AND (NOT condition2 OR condition3);
```

DML / DELETE

- It is a statement used to update specific rows in a table.

SYNTAX: `DELETE FROM table_name WHERE ...;`

DB BROWSER

```
1 DELETE FROM order_item
2 WHERE order_item_id=5;
3
4
5
6
```

Adding three products to the `order_item` table.

NOTE: Always include the WHERE clause, otherwise multiple data can be deleted accidentally.

DML / DELETE

Account

AID	FirstName	LastName	Username	BirthDate	Role
1	Juan	Dela Cruz	OneDy	2000-09-11	0
2	Maria	Sta. Ana	lilLamb	1999-06-29	0

AID	FirstName	LastName	Username	BirthDate	Role
2	Maria	Sta. Ana	admin01	1999-06-29	1

DML / DELETE

Account

AID	FirstName	LastName	Username	BirthDate	Role
1	Juan	Dela Cruz	OneDy	2000-09-11	0
2	Maria	Sta. Ana	lilLamb	1999-06-29	0

AID	FirstName	LastName	Username	BirthDate	Role
2	Maria	Sta. Ana	admin01	1999-06-29	1

```
DELETE FROM Account
WHERE AID=1;
```

Basic Data Retrieval

DML / SELECT

- It is a statement used to retrieve data from one or more tables.

SYNTAX:
```
SELECT column1, column2,.., columnN FROM table_name
WHERE condition1 AND (NOT condition2 OR condition3);
```

DML / SELECT

Account

AID	FirstName	LastName	Username	BirthDate	Role
1	Juan	Dela Cruz	OneDy	2000-09-11	0
2	Maria	Sta. Ana	lilLamb	1999-06-29	0
3	Pedro	Malaya	peter20	1998-01-03	1
4	Elias	Pinaglabanan	elyuzzz	2002-08-17	0
5	Mateo	Abante	10math01	2000-03-22	1

AID	FirstName	LastName	Username	BirthDate	Role
3	Pedro	Malaya	peter20	1998-01-03	1
5	Mateo	Abante	10math01	2000-03-22	1

```
SELECT * FROM Account WHERE Role=1;
```

DML / SELECT

Account

AID	FirstName	LastName	Username	BirthDate	Role
1	Juan	Dela Cruz	OneDy	2000-09-11	0
2	Maria	Sta. Ana	lilLamb	1999-06-29	0
3	Pedro	Malaya	peter20	1998-01-03	1
4	Elias	Pinaglabanan	elyuzzz	2002-08-17	0
5	Mateo	Abante	10math01	2000-03-22	1

AID	FirstName	LastName	Username	BirthDate	Role
2	Maria	Sta. Ana	lilLamb	1999-06-29	0
3	Pedro	Malaya	peter20	1998-01-03	1

```
SELECT * FROM Account WHERE BirthDate<“2020-01-01”;
```

DML / SELECT

Account

AID	FirstName	LastName	Username	BirthDate	Role
1	Juan	Dela Cruz	OneDy	2000-09-11	0
2	Maria	Sta. Ana	lilLamb	1999-06-29	0
3	Pedro	Malaya	peter20	1998-01-03	1
4	Elias	Pinaglabanan	elyuzzz	2002-08-17	0
5	Mateo	Abante	10math01	2000-03-22	1

FirstName	LastName
Pedro	Malaya

```
SELECT FirstName, LastName FROM Account WHERE AID=3;
```

DML / SELECT

Account

AID	FirstName	LastName	Username	BirthDate	Role
1	Juan	Dela Cruz	OneDy	2000-09-11	0
2	Maria	Sta. Ana	lilLamb	1999-06-29	0
3	Pedro	Malaya	peter20	1998-01-03	1
4	Elias	Pinaglabanan	elyuzzz	2002-08-17	0
5	Mateo	Abante	10math01	2000-03-22	1

Username
OneDy
lilLamb
elyuzzz

```
SELECT Username FROM Account WHERE Role=0;
```

DML / SELECT

sales

sales_id	order_id	total_amount	sales_date
1	1	199.99	2023-08-11
2	3	2075.32	2023-08-19
3	4	548.00	2023-09-21
4	7	1200.01	2023-09-23
5	11	499.98	2023-09-26

```
SELECT order_id FROM sales WHERE total_amount<1000;
```

DML / SELECT

sales

sales_id	order_id	total_amount	sales_date
1	1	199.99	2023-08-11
2	3	2075.32	2023-08-19
3	4	548.00	2023-09-21
4	7	1200.01	2023-09-23
5	11	499.98	2023-09-26

```
SELECT order_id FROM sales WHERE total_amount<=1000;
```

order_id
1
4
11

DML / SELECT

inventory

inventory_id	product_id	unit_price	date_added
1	1	288.98	2023-08-14
2	3	1044.00	2023-08-22
3	4	548.49	2023-09-03
4	7	59.22	2023-09-07
5	11	799.80	2023-09-11

```
SELECT product_id FROM inventory
WHERE unit_price BETWEEN 500 AND 1000;
```

product_id
4
11

DML / SELECT

product

product_id	category_id	product_name	product_description	date_created
1	18	QuantumGlow Smart Lamp	Illuminate your space …	2023-07-02
2	21	EcoHarmony Bamboo Travel Mug	Sip sustainably with the …	2023-07-02
3	9	TechVibe Wireless Earbuds	Immerse yourself in the …	2023-07-03
4	1	EverFresh Herb Keeper	Keep your herbs fresh for longer …	2023-07-03
5	11	AdventurePro Multi-Tool Kit	Be ready for anything with the …	2023-07-03

```
SELECT product_id FROM product
WHERE product_name LIKE “%ar%”;
```

product_id
1
2
3

DML / SELECT

product

product_id	category_id	product_name	product_description	date_created
1	18	QuantumGlow Smart Lamp	Illuminate your space …	2023-07-02
2	21	EcoHarmony Bamboo Travel Mug	Sip sustainably with the …	2023-07-02
3	9	TechVibe Wireless Earbuds	Immerse yourself in the …	2023-07-03
4	1	EverFresh Herb Keeper	Keep your herbs fresh for longer …	2023-07-03
5	11	AdventurePro Multi-Tool Kit	Be ready for anything with the …	2023-07-03

```
SELECT product_id FROM product
ORDER BY product_name ASC, date_created DESC
LIMIT 3 OFFSET 0;
```

product_id
5
2
4

PAGE: numbers 1, 2, 3, 4…
LIMIT ***l*** = any positive numbers: 5, 25, 50, 100…
OFFSET *o* = (PAGE - 1) * LIMIT

Data Summarization

Data Summarization / DISTINCT

product

product_id	category_id	product_name	product_description	date_created
1	18	QuantumGlow Smart Lamp	Illuminate your space …	2023-07-02
2	21	EcoHarmony Bamboo Travel Mug	Sip sustainably with the …	2023-07-02
3	9	TechVibe Wireless Earbuds	Immerse yourself in the …	2023-07-03
4	1	EverFresh Herb Keeper	Keep your herbs fresh for longer …	2023-07-03
5	11	AdventurePro Multi-Tool Kit	Be ready for anything with the …	2023-07-03

```
SELECT DISTINCT category_id FROM product;
```

category_id
18
21
9
1
11

Data Summarization / SUM

sales

sales_id	order_id	total_amount	sales_date
1	1	199.99	2023-08-11
2	3	2075.32	2023-08-19
3	4	548.00	2023-09-21
4	7	1200.01	2023-09-23
5	11	499.98	2023-09-26

```
SELECT SUM(total_amount) AS grand_total FROM sales;
```

grand_total
4523.30

Data Summarization / AVG

sales

sales_id	order_id	total_amount	sales_date
1	1	199.99	2023-08-11
2	3	2075.32	2023-08-19
3	4	548.00	2023-09-21
4	7	1200.01	2023-09-23
5	11	499.98	2023-09-26

```
SELECT AVG(total_amount) AS average_sales FROM sales;
```

average_sales
904.66

Data Summarization / MIN

sales

sales_id	order_id	total_amount	sales_date
1	1	199.99	2023-08-11
2	3	2075.32	2023-08-19
3	4	548.00	2023-09-21
4	7	1200.01	2023-09-23
5	11	499.98	2023-09-26

```
SELECT MIN(total_amount) AS lowest_sales FROM sales;
```

lowest_sales
199.99

Data Summarization / MAX

sales

sales_id	order_id	total_amount	sales_date
1	1	199.99	2023-08-11
2	3	2075.32	2023-08-19
3	4	548.00	2023-09-21
4	7	1200.01	2023-09-23
5	11	499.98	2023-09-26

```
SELECT MAX(total_amount) AS lowest_sales FROM sales;
```

highest_sales
2075.32

Data Summarization / COUNT

sales

sales_id	order_id	total_amount	sales_date
1	1	199.99	2023-08-11
2	3	2075.32	2023-08-19
3	4	548.00	2023-09-21
4	7	1200.01	2023-09-23
5	11	499.98	2023-09-26

```
SELECT COUNT(sales_id) AS number_of_sales FROM sales;
```

number_of_sales
5

Data Summarization

sales

sales_id	order_id	total_amount	sales_date
1	1	199.99	2023-08-11
2	3	2075.32	2023-08-19
3	4	548.00	2023-09-21
4	7	1200.01	2023-09-23
5	11	499.98	2023-09-26

```
SELECT
    strftime("%Y-%m", sales_date) AS month,
    SUM(total_amount) AS sub_total
FROM sales ORDER BY month GROUP BY month;
```

month	sub_total
2023-08	2275.31
2023-09	2247.99

Summary

- **Structured Query Language** (SQL) is a specialized programming language used to query, manipulate, and manage relational databases.
- Data Definition Language includes `CREATE`, `ALTER`, and `DROP`.
- Data Manipulation Language, includes `SELECT`, `INSERT`, `UPDATE`, and `DELETE`.
- `SELECT` is used to query records from tables.
- Data summarization functions can be used to aggregate data from a table.

QUESTIONS?

THANK YOU!

References

- Elmasri, R. et al. (2016). Fundamentals of Database Systems, 7th ed. Pearson Higher Education, 221 River Street, Hoboken, NJ 07030.