ជំពូកទី ៥

Structured Query Language

1. Introduction to SQL

ក្នុងជំពូកនេះយើងនឹងសិក្សាអំពីមូលដ្ឋានគ្រឹះនៃ relational model's standard language ដែលគេហៅថា **Structured Query Language (SQL)**។ ភាសាផ្សំឡើងដោយ commands ជាច្រើនដើម្បីអោយ user មានលទ្ធភាពបង្កើត database និង table structures ព្រមទាំងដំណើរការប្រភេទជាច្រើននៃ data manipulation និង data administration ហើយស្វែងរក (query) ព័ត៌មានមានប្រយោជន៍ចេញពី database។ RDBMS (Relational Database Management System) ទាំងអស់សុទ្ធតែទ្រទ្រង់ SQL ហើយក៏មាន software vendors មួយចំនួនកំពុងតែអភិវឌ្ឍន៍បន្ថែមទៀតទៅលើ basic SQL commands។ **SQL** គឺជា non-procedural language មានន័យថា user គ្រាន់តែកំនត់អ្វីដែលត្រូវការ មិនចាំបាច់កំនត់ថាតើវាត្រូវដំណើរការដោយរបៀបណាទេ។ នៅពេលប្រើប្រាស់ SQL commands, end users មិនចាំបាច់ដឹងពី physical data storage format និងសកម្មភាពសាំញ៉ាំកើតឡើងនៅពេល SQL command ដំណើរការ។

គោលបំណងនៃ SQL គឺ:

- បង្កើត database and relation structures។
- ដំណើរការ basic data management tasks ដូចជា insertion, modification, and deletion ទិន្នន័យពី relation។
- ដំណើរការស្វែងរកព័ត៌មានដែលមានលក្ខណៈងាយស្រួល និងសុគស្ទាញដើម្បីបំលែង raw data ទៅជាព័ត៌មាន។

លើសពីនេះទៀត command structure and syntax នៃ database language ត្រូវមានលក្ខណៈងាយស្រួលក្នុងការសិក្សា និងចុងបញ្ចប់ត្រូវតែ portable មានន័យថាវាត្រូវតែគោរពតាម standard ហេតុដូចនេះយើងនៅតែអាចប្រើប្រាស់ command structure and

syntax ដូចគ្នានៅពេលធ្វើការផ្លាស់ប្តូរពី DBMS មួយទៅកាន់ DBMS មួយទៀត។ SQL ជាឧទាហរណ៍មួយនៃ **transform-oriented language** វីជាភាសាដែលប្រើប្រាស់ relations សំរាប់បំលែងពី inputs អោយទៅជា outputs។ SQL ចែកជា 2 ប្រភេទចំបងគឺ:

- Data Definition Language (DDL) សំរាប់បង្កើតរចនាសម្ព័ន្ធ database។
- Data Manipulation Language (DML) សំរាប់ទទួល និងកែប្រែទិន្នន័យ។

SQL មិនបានផ្តុក flow of control commands ហើយម្យ៉ាងវិញទៀតវាគ្មាន IF...THEN...ELSE, GO TO, DO...WHILE រឺ commands ផ្សេងទៀតផ្តល់នូវ flow of controls ទេ។

History of SQL

Relational model ត្រូវបានបង្កើតដោយ E.F.Codd នៅក្នុងឆ្នាំ 1970 នៅពេលដែលបានធ្វើការចោះពុម្ពផ្សាយជាសារធារណៈ ហើយនៅក្នុងឆ្នាំ 1974, D. Chamberlin មកពី IBM San José Laboratory ផងដែរបានបង្កើតភាសាមួយហៅថា 'Structured English Query Language' or SEQUEL។ បន្ទាប់មកនៅឆ្នាំ 1976 មានការបន្ថែមបន្ថយទៅលើវា (SEQUEL/2) ហើយបានកែប្រែឈ្មោះទៅជា SQL វិញ ហើយមានមនុស្សភាគច្រើនអានថា 'see-quel' តែតាមពិតការអានជាផ្លូវការគឺ 's-q-l'។ បន្ទាប់មក IBM បង្កើត prototyping DBMS ហៅថា System R ដែលមានមូលដ្ឋានគ្រឹះលើ SEQUEL/2 ហើយបន្ទាប់មកទៀតនៅចុងទសវត្ស 1970 ORACLE Corporation បានបង្កើត commercial relational DBMS ដំបូងគេដែលមានមូលដ្ឋានគ្រឹះលើ SQL។ ឃើញដូច្នោះ INGRES ក៏បានបង្កើត relational DBMS ផងដែរដោយមានមូលដ្ឋានគ្រឹះលើ query language ហៅថា QUEL។ បន្ទាប់ពី SQL ត្រូវបានគេចាត់ទុកជា standard database language for relational system ទើប INGRES បំលែងទៅជា SQL-based DBMS វិញ។ នៅក្នុងឆ្នាំ 1992 ដោយមានការកែសំរួលជាចំបងទៅលើ ISO standard ហើយគេហៅថា SQL 2 or SQL 92។ បន្ទាប់មកមានការកែប្រែបន្ថែមទៀតទៅលើវា ដោយបន្ថែមលក្ខណៈ object-oriented management ទើបគេហៅថា SQL 3។

Writing SQL Commands

SQL statement ផ្តុកនូវ **reserved words** និង **user-defined words**។ Reserved words គឺជាផ្នែកកំនត់ជាក់លាក់នៅក្នុង SQL language ហើយមានអត្ថន័យផងដែរ។

User-defined words គឺជាពាក្យបង្កើតដោយ user ហើយជាទូទៅតែងតែបង្ហាញពី database objects ដូចជា relations, columns, views, indexes, and so on។

SQL statements មានលក្ខណៈ **case-insensitive** មានន័យថាវាមិនប្រកាន់តួអក្សរតូច រីធំទេគឺយើងអាចសរសេរជាអក្សរតូចទាំងអស់ រីអក្សរតូចលាយអក្សរធំ រីអក្សរធំទាំងអស់ក៏បានហើយសញ្ញា semicolon " ; " ប្រើសំរាប់បញ្ចាក់អំពីទីបញ្ចប់នៃ SQL statement តែយើងចង់សរសេរក៏បានមិនសរសេរក៏បាន។ មានការកំនត់សំគាល់មួយចំនួនចំពោះ SQL statements:

- តួអក្សរធំប្រើសំរាប់បង្ហាញ reserved words
- តួអក្សរតូចប្រើសំរាប់បង្ហាញ user-defined words
- Vertical bar '|' បញ្ជាក់ពីជំរើស (choice) ឧទាហរណ៍: A | B | C
- Curly braces '{}' បញ្ជាក់ពីធាតុត្រូវការ (required element) ឧទាហរណ៍: {a}
- Square brackets '[]' បញ្ជាក់ពីធាតុមិនចាំបាច់ (optional element) ឧទាហរណ៍: [a]
- Ellipsis (...) បញ្ជាក់ពី optional repetition of item zero or more times ឧទាហរណ៍: {a | b} [,c...]

2. Data Manipulation Language

Data Manipulation Language ជាភាសាវងមួយដែលប្រើប្រាស់ statements ដូចជា:

- SELECT ស្វែងរកទិន្នន័យនៅក្នុង database។
- INSERT បន្ថែមទិន្នន័យទៅកាន់ table។
- UPDATE កែប្រែទិន្នន័យនៅក្នុង table។
- DELETE លុបទិន្នន័យចេញពី table។

2. 1. Simple Queries

គោលបំណងសំខាន់នៃ SELECT statement គឺដើម្បីទទួល និងបង្ហាញទិន្នន័យចេញពី database tables មួយ រីច្រើន។ SELECT គឺជា SQL command ដែលប្រើប្រាស់ញឹកញាប់ជាងគេ ហើយដែលមានទំរង់ទូទៅដូចខាងក្រោម។ *column_expression* បង្ហាញពី column name or expression, *table_name* គឺជាឈ្មោះ tables រឺ views ហើយ *alias* គឺជាពាក្យសរសេរកាត់ចំពោះឈ្មោះ table។

```
SELECT [DISTINCT | ALL] {* | [column_expression [AS new_name]][,...]}
FROM table_name [alias] [,...]
[WHERE condition]
[GROUP BY column_list]
[HAVING condition]
[ORDER BY column_list]
```

ខាងក្រោមនេះជាលំដាប់ដំណើរការរបស់ SQL statement:

FROM កំនត់ឈ្មោះ tables មួយ ឬច្រើនសំរាប់ប្រើប្រាស់

WHERE ចំរាញ់យក rows ណាដែលបំពេញតាមលក្ខខណ្ឌ

GROUP BY បង្កើតក្រុមនៃ rows នៅក្នុង column តែមួយ

HAVING ចំរាញ់យកក្រុមណាដែលបំពេញតាមល័ក្ខខ័ណ្ឌ

SELECT កំនត់ columns ណាខ្លះដែលត្រូវបង្ហាញក្នុងលទ្ធផល

ORDER BY កំនត់លំដាប់នៃការបង្ហាញលទ្ធផល

លំដាប់នៃ SQL statement មិនអាចធ្វើការផ្លាស់ប្តូរបានទេ ហើយជាទូទៅ SQL statement តែងតែមាន SELECT និង FROM ហើយចំពោះផ្នែកនៅសល់អាចអត់ក៏បាន។

ឧទាហរណ៍ 1: (Retrieve all columns, all rows) បង្ហាញព័ត៌មានលំអិតបុគ្គលិកទាំងអស់?

យើងឃើញថា គ្មានការដាក់កំរិតក្នុង query នេះទេដូច្នេះមិនចាំបាច់មាន WHERE clause ទេ ហើយ columns ទាំងអស់សុទ្ធតែត្រូវបង្ហាញ។

```
SELECT sno, fname, lname, address, tel_no, position, sex, dob,
salary, nin, bno
FROM staff;
```

ឬ

```
SELECT * FROM staff;
```

Sno	FName	LName	Address	Tel_No	Position	Sex	DOB	Salary	NIN	Bno
SL21	John	White	19 Taylor St, Cranford, London	0171-884-5112	Manager	F	1-Oct-45	30000	WK442011B	B5
SG37	Ann	Beech	81 George St, Glasgow PA1 2JR	0141-848-3345	Snr Asst	F	10-Nov-60	12000	WL432514C	B3
SG14	David	Ford	63 Ashby St, Partick, Glasgow G11	0141-339-2177	Deputy	M	24-Mar-58	18000	WL220658D	B3
SA9	Mary	Howe	2 Elm Pl, Aberdeen AB2 3SU		Assistant	F	19-Feb-70	9000	WM532187D	B7
SG5	Susan	Brand	5 Gt Western Rd, Glasgow G12	0141-334-2001	Manager	F	3-Jun-40	24000	WK588932E	B3
SL41	Julie	Lee	28 Malvern St, Kilburn NW2	0181-554-3541	Assistant	F	13-Jun-65	9000	WA290573K	B5

(រូប 5.1)

សញ្ញាផ្កាយ “*” តំណាងអោយគ្រប់ columns ហើយលទ្ធផលបង្ហាញក្នុងរូប 5.1។

ឧទាហរណ៍ 2: (Retrieve specific columns, all rows) បង្ហាញប្រាក់ខែបុគ្គលិកទាំងអស់ដោយយកតែ staff number, first and last name និង salary មកដាក់ក្នុងលទ្ធផល?

```
SELECT sno, fname, lname, salary
FROM staff;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.2

Sno	FName	LName	Salary
SL21	John	White	30000
SG37	Ann	Beech	12000
SG14	David	Ford	18000
SA9	Mary	Howe	9000
SG5	Susan	Brand	24000
SL41	Julie	Lee	9000

(រូប 5.2)

ឧទាហរណ៍ 3: (Use of DISTINCT) បង្ហាញ property number នៃ properties ទាំងអស់ដែលបានមើល?

```
SELECT pno
FROM viewing;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.3a

គួរកត់សំគាល់ផងដែរថាមានការលេចឡើងនូវតំលៃស្ទួនជាច្រើន នៅពេលដែលយើងធ្វើចំណោល (projection) ដូចនេះដើម្បីលុបតំលៃស្ទួននេះចេញពីលទ្ធផល គេប្រើប្រាស់ DISTINCT keyword។

```
SELECT DISTINCT pno
FROM viewing;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.3b

pno
PA14
PG4
PG4
PA14
PG36

(រូប 5.3a)

pno
PA14
PG4
PG36

(រូប 5.3b)

ឧទាហរណ៍ 4: (Calculated fields) បង្ហាញប្រាក់ខែ (monthly salary) នៃបុគ្គលិកទាំងអស់ដោយយកតែ staff number, first and last name និង salary មកដាក់ក្នុងលទ្ធផល?

```
SELECT sno, fname, lname, salary/12
FROM staff;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.4

Sno	FName	LName	col4
SL21	John	White	2500.00
SG37	Ann	Beech	1000.00
SG14	David	Ford	1500.00
SA9	Mary	Howe	750.00
SG5	Susan	Brand	2000.00
SL41	Julie	Lee	750.00

(រូប 5.4)

Calculated field (រឺ computed or derived field) ទាក់ទងនឹង expression ដែលប្រើប្រាស់ addition, subtraction, multiplication, division និងអនុគមន៍មួយចំនួនផ្សេងទៀត។ លើសពីនេះទៀត calculated column អាចកើតឡើងពីការចូលរួមរវាង table columns លើសពី 1។

ប្រសិនបើយើងក្រលេកទៅមើលលទ្ធផលក្នុងរូប 5.4 យើងឃើញថា column ទី 4 បង្ហាញឈ្មោះជា *col4*។ ជាធម្មតា column នៅក្នុងលទ្ធផលអាចទាញយកឈ្មោះ column ត្រូវគ្នាក្នុង database table នោះ ប៉ុន្តែក្នុងករណីនេះ SQL មិនដឹងថាតើត្រូវដាក់ឈ្មោះ column នោះយ៉ាងដូចម្តេច ទើបផ្តល់ឈ្មោះតាមរយៈទីតាំង column នោះ។ ក្នុងករណីនេះប្រសិនបើចង់ប្តូរឈ្មោះ (column heading) column ទី 4 ទៅជា ‘monthly_salary’ គេត្រូវប្រើ AS clause ដូចខាងក្រោម

```
SELECT sno, fname, lname, salary/12 AS monthly_salary
FROM staff;
```

Row selection (WHERE clause)

ឧទាហរណ៍ SELECT statements មួយចំនួនខាងលើសុទ្ធតែទទួល rows ទាំងអស់ពី table។ ទោះបីជាយ៉ាងណាក៏ដោយ យើងតែងតែធ្វើការកំនត់ជ្រើសរើសយក rows ដែលត្រូវការប៉ុណ្ណោះមកដាក់ក្នុងលទ្ធផល ដូចនេះទាមទារការប្រើប្រាស់ WHERE clause។ ល័ក្ខខ័ណ្ឌស្វែងរកជាមូលដ្ឋានចែកជា 5 ប្រភេទ (5 basic search conditions) គឺ Comparison, Range, Set membership, Pattern match និង Null។

- **Comparison** ប្រើសំរាប់ប្រៀបធៀបតំលៃ expression មួយជាមួយតំលៃ expression មួយទៀត។ នៅក្នុង SQL មាន comparison operators មួយចំនួនដូចជា =, <, >, <=, >=. <>, != និង logical operators ដូចជា **AND**, **OR**, and **NOT** ព្រមទាំងវង់ក្រចកប្រសិនបើចាំបាច់។

ឧទាហរណ៍ 5: បង្ហាញបុគ្គលិកដែលមានប្រាក់ខែលើស $10,000?

```
SELECT sno, fname, lname, position, salary
FROM staff
WHERE salary > 10000;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.5

Sno	FName	LName	position	Salary
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Snr Asst	12000.00
SG14	David	Ford	Deputy	18000.00
SG5	Susan	Brand	Manager	24000.00

(រូប 5.5)

ឧទាហរណ៍ 6: បង្ហាញ branch offices ដែលអាស័យដ្ឋាននៅ London រឺ Glasgow?

```
SELECT bno, street, area, city, post_code
FROM branch
WHERE city = 'London' OR city = 'Glasgow';
```

លទ្ធផលបង្ហាញក្នុងរូប 5.6

Bno	Street	Area	City	Post_Code
B5	22 Deer Rd	Sidcup	London	SW1 4EH
B3	163 Main St	Partick	Glasgow	G11 9QX
B2	56 Clover Dr		London	NW10 6EU

(រូប 5.6)

- **Range** សំរាប់ត្រួតពិនិត្យថាតើតំលៃ expression ស្ថិតក្នុងចន្លោះតំលៃកំនត់ដែរឬអត់ ហើយក្នុង SQL គេប្រើ BETWEEN...AND.../NOT BETWEEN...AND... ។

ឧទាហរណ៍ 7: បង្ហាញបុគ្គលិកដែលមានប្រាក់ខែចន្លោះពី $20,000 ដល់ $30,000?

```
SELECT sno, fname, lname, position, salary
FROM staff
WHERE salary BETWEEN 20000 AND 30000;
```

រឺ

```
SELECT sno, fname, lname, position, salary
FROM staff
WHERE salary >= 20000 AND salary <= 30000;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.7

Sno	FName	LName	position	Salary
SL21	John	White	Manager	30000.00
SG5	Susan	Brand	Manager	24000.00

(រូប 5.7)

- **Set membership** សំរាប់ត្រួតពិនិត្យថាតើតំលៃ expression ស្មើនឹងតំលៃមួយក្នុងចំណោមសំនុំតំលៃដែរ រីអត់ ហើយក្នុង SQL គេប្រើ IN/NOT IN។

ឧទាហរណ៍ 8: បង្ហាញបុគ្គលិកណា ដែលមានតួនាទីជា Managers និង Deputy Managers?

```
SELECT sno, fname, lname, position
FROM staff
WHERE position IN ('Manager','Deputy');
```

រឺ

```
SELECT sno, fname, lname, position
FROM staff
WHERE position = 'Manager' or position = 'Deputy';
```

លទ្ធផលបង្ហាញក្នុងរូប 5.8

Sno	FName	LName	position
SL21	John	White	Manager
SG14	David	Ford	Deputy
SG5	Susan	Brand	Manager

(រូប 5.8)

- **Pattern match** សំរាប់ត្រួតពិនិត្យថាតើ string ត្រូវជាមួយគំរូកំនត់ដែរ រីអត់ ហើយក្នុង SQL គេប្រើ LIKE/NOT LIKE។ ជាពិសេសគេប្រើនៅពេលដែលយើងធ្វើការស្វែងរកទិន្នន័យណាដែលយើងពុំដឹងច្បាស់លាស់ រីពុំដឹងពិតប្រាកដ។ មាននិមិត្តសញ្ញាមួយចំនួនប្រើប្រាស់ជាមួយ LIKE ដូចជា:

- Percent character (%): តំណាងអោយអក្សរគ្មានរីក៏ច្រើន (Ms-Access ប្រើ *)
- Underscore character (_): តំណាងអោយមួយតួអក្សរ (Ms-Access ប្រើ ?)

ឧទាហរណ៍:

Address LIKE "H%" មានន័យថាស្វែងរកអាស័យដ្ឋានចាប់ផ្តើមដោយអក្សរ 'H' ហើយតួអក្សរខាងក្រោយជាតួអក្សរអ្វីក៏បាន យ៉ាងហោចណាស់មួយតួអក្សរ។

Address LIKE "%e" មានន័យថាស្វែងរកអាស័យដ្ឋានបញ្ចប់ដោយអក្សរ 'e' ហើយតួអក្សរខាងមុខអាចជាតួអក្សរអ្វីក៏បាន យ៉ាងហោចណាស់មួយតួអក្សរ។

Address LIKE "%Glasgow%" មានន័យថាស្វែងរកអាស័យដ្ឋានដែលមានពាក្យ 'Glasgow' ទោះនៅទីតាំងណាក៏ដោយ។

Address LIKE "_ _ _" មានន័យថាស្វែងរកអាស័យដ្ឋានដែលមាន 3 តួអក្សរ។

Address LIKE "_ _ a" មានន័យថាស្វែងរកអាស័យដ្ឋានដែលមាន 3 តួអក្សរ ហើយបញ្ចប់ដោយតួអក្សរ 'a'។

Address NOT LIKE "H%" មានន័យថាស្វែងរកអាស័យដ្ឋានដែលមានតួអក្សរទីមួយខុសពីអក្សរ 'H' និងតួអក្សរខាងក្រោយជាតួអក្សរអ្វីក៏បាន យ៉ាងហោចណាស់មួយតួអក្សរ។

ឧទាហរណ៍ 9: បង្ហាញបុគ្គលិកដែលមាន string 'Glasgow' នៅក្នុង address column?

```
SELECT sno, fname, lname, address, salary
FROM staff
WHERE address LIKE '%Glasgow%';
```

លទ្ធផលបង្ហាញក្នុងរូប 5.9

Sno	FName	LName	Address	Salary
SG37	Ann	Beech	81 George St, Glasgow PA1 2JR	12000
SG14	David	Ford	63 Ashby St, Partick, Glasgow G11	18000
SG5	Susan	Brand	5 Gt Western Rd, Glasgow G12	24000

(រូប 5.9)

- **Null** សំរាប់ត្រួតពិនិត្យថាតើ column មានផ្ទុកតំលៃ null ដែរឬអត់ ហើយក្នុង SQL គេប្រើ IS NULL/IS NOT NULL។

ឧទាហរណ៍ 10: បង្ហាញព័ត៌មានលំអិតចំពោះការមើល properties PG4 ទាំងអស់ដែលមានមិនបានផ្តល់ comment?

```
SELECT rno, date
FROM viewing
WHERE pno = 'PG4' AND comment IS NULL;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.10

Rno	Date
CR56	26-May-98

(រូប 5.10)

2. 2. Sorting Results (ORDER BY clause)

ជាទូទៅលទ្ធផលទទួលបានពីដំណើរការស្នើសុំ គឺមិនទាន់បានតំរៀបទៅតាមទំរង់ណាមួយនៅឡើយទេ ទោះបីជាយ៉ាងណាក៏ដោយយើងអាចតំរៀប (sort) លទ្ធផលទទួលបាននោះដោយប្រើ ORDER BY clause នៅក្នុង SELECT statement។ ORDER BY clause ផ្ទុកនូវបណ្តុំ **column identifiers** ផ្តាច់គ្នាដោយសញ្ញា commas ',' ហើយ column identifier អាចជា column name or column number សំរាប់កំនត់សមាសធាតុនៃ SELECT list ដោយលេខ

ទីតាំងចាប់ផ្តើមពីលេខ 1 ដោយគិតពីខាងឆ្វេង។ គេប្រើប្រាស់ column number នៅពេលដែលយើងចង់តំរៀបតាម expression គ្មាន AS clause ដើម្បីសំគាល់ឈ្មោះ។ ORDER BY clause អាចតំរៀប records តាម 2 ទំរង់គឺលំដាប់កើន Ascending (ASC) ជា default និងលំដាប់ចុះ Descending (DESC)។

ឧទាហរណ៍ 11: បង្ហាញប្រាក់ខែបុគ្គលិកទាំងអស់ដោយយកតែ staff number, first and last name, and salary មកដាក់ក្នុងលទ្ធផលហើយតំរៀបតាមប្រាក់ខែខ្ពស់មកទាប?

```
SELECT sno, fname, lname, salary
FROM staff
ORDER BY salary DESC;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.11

Sno	FName	LName	Salary
SL21	John	White	30000
SG5	Susan	Brand	24000
SG14	David	Ford	18000
SG37	Ann	Beech	12000
SA9	Mary	Howe	9000
SL41	Julie	Lee	9000

(រូប 5.11)

ឧទាហរណ៍ 12: បង្ហាញព័ត៌មាន property ដោយយកតែ property number, type, rooms, and rent មកដាក់ក្នុងលទ្ធផលហើយតំរៀបតាមប្រភេទ (type)

```
SELECT pno, type, rooms, rent
FROM property_for_rent
ORDER BY type;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.12a

Pno	Type	Rooms	Rent
PL94	Flat	4	400
PG4	Flat	3	350
PG36	Flat	3	375
PG16	Flat	4	450
PA14	House	6	650
PG21	House	5	600

(រូប 5.12a)

ប្រសិនបើយើងបន្ថែម rent column មួយទៀតទៅកាន់ ORDER BY clause យើងសង្កេតឃើញថា លទ្ធផលមានការប្រែប្រួលព្រោះពេលនេះតំរៀបទៅតាម type សិនហើយបានវាយក type group នីមួយៗទៅតំរៀបតាមរយៈ rent ម្តងទៀត។ លទ្ធផលទទួលបានដូចរូប 5.12b។

```
SELECT pno, type, rooms, rent
FROM property_for_rent
ORDER BY type, rent DESC;
```

Pno	Type	Rooms	Rent
PG16	Flat	4	450
PL94	Flat	4	400
PG36	Flat	3	375
PG4	Flat	3	350
PA14	House	6	650
PG21	House	5	600

(រូប 5.12b)

2. 3. Using the SQL Aggregate Functions

The ISO standard បានកំនត់ aggregate functions ចំនួន 5 គឺ:

- COUNT (A) : សំរាប់រាប់ចំនួនតំលៃក្នុង Column A ដែលមានតំលៃមិន NULL ។
- SUM (A) : សំរាប់គណនាផលបូកតំលៃទាំងអស់ក្នុង Column A ។
- AVG (A) : សំរាប់គណនាមធ្យមភាគតំលៃទាំងអស់នៅក្នុង Column A ។
- MAX (A) : សំរាប់តំលៃធំបំផុតនៃ Column A ។
- MIN (A) : សំរាប់តំលៃតូចបំផុតនៃ Column A ។

Functions ទាំងនេះដំណើរការលើ single column of a table ហើយទទួលបានន្ូវតំលៃតែមួយគត់។ COUNT, MIN និង MAX ធ្វើការជាមួយ numeric and non-numeric columns ប៉ុន្តែ SUM និង AVG ធ្វើការជាមួយតែ numeric columns ។ COUNT(*) គឺជាការប្រើប្រាស់ជាពិសេសទៅលើ COUNT ដើម្បីរាប់ចំនួន rows ទាំងអស់មានក្នុង table ។ នៅពេលដំណើរការ functions ទាំងអស់លើកលែងតែ COUNT(*) ធ្វើការលុបតំលៃ nulls ជាមុនរួចទើបធ្វើការជាមួយតំលៃខុសពី nulls ដែលនៅសល់។

គួរកត់សំគាល់ផងដែរថា គេអាចប្រើ aggregate functions នៅ SELECT និង HAVING clauses តែប៉ុណ្ណោះ។ ប្រសិនបើ SELECT list មានបញ្ចូល aggregate functions និងគ្មាន GROUP BY clause មានន័យថាវាធ្វើការទៅលើទិន្នន័យក្នុង table ទាំងមូលដោយចាត់ទុកជា 1 ក្រុម។ ឧទាហរណ៍: Query ខាងក្រោមនេះមិនត្រឹមត្រូវព្រោះមានបញ្ចូល sno column

```
SELECT sno, COUNT(salary)
FROM staff;
```

ឧទាហរណ៍ 13: តើមាន property ចំនួនប៉ុន្មានដែលមានថ្លៃជួលលើសពី $350? (How many properties cost more than $350 per month to rent?)

```
SELECT COUNT(*) AS count
FROM property_for_rent
WHERE rent > 350;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.13

count
5

(រូប 5.13)

ឧទាហរណ៍ 14: តើមាន property ចំនួនប៉ុន្មានដែលបានមើលក្នុងខែឧសភា 1998? (How many different properties were viewed in May 1998?)

```
SELECT COUNT(pno) AS count
FROM viewing
WHERE date BETWEEN '1-May-98' AND '31-May-98';
```

លទ្ធផលបង្ហាញក្នុងរូប 5.14

count
3

(រូប 5.14)

ឧទាហរណ៍ 15: ស្វែងរកចំនួន Managers និងប្រាក់ខែសរុបរបស់ពួកគេ? (Find the total number of Managers and the sum of their salaries)

```
SELECT COUNT(sno) AS count, SUM(salary) AS sum
FROM staff
WHERE position = 'Manager';
```

លទ្ធផលបង្ហាញក្នុងរូប 5.15

count	sum
2	54000.00

(រូប 5.15)

ឧទាហរណ៍ 16: ស្វែងរកប្រាក់ខែបុគ្គលិកតិចជាងគេ, ច្រើនជាងគេ និងជាមធ្យម? (Find the minimum, maximum, and average staff salary)

```
SELECT MIN(salary) AS min, MAX(salary) AS max, AVG(salary)
       AS avg
FROM staff
```

លទ្ធផលបង្ហាញក្នុងរូប 5.16

min	max	avg
9000.00	30000.00	17000.00

(រូប 5.16)

2. 4. Grouping Results (GROUP BY clause)

Summary queries ខាងលើនេះប្រហាក់ប្រហែលទៅនឹង totals នៅផ្នែកខាងក្រោមនៃ reports ទោះបីជាយ៉ាងណាក៏ដោយក៏មាន subtotals ប្រើប្រាស់ក្នុង reports ដែលទាមទារអោយប្រើប្រាស់ GROUP BY clause ដើម្បីផ្គុំទិន្នន័យជាក្រុមៗ។ Query មានបញ្ជូល GROUP BY clause គេហៅថា **grouped query** ព្រោះវាប្រមូលផ្គុំទិន្នន័យជាក្រុមៗ ហើយផ្តល់នូវ row សង្ខេបមួយសំរាប់ក្រុមនីមួយៗ ហើយ columns ប្រើប្រាស់នៅក្រោយ ORDER BY clause គេអោយឈ្មោះថា **grouping columns**។ រាល់ columns ដែលមាននៅក្នុង SELECT list ត្រូវតែបង្ហាញក្នុង GROUP BY clause ប្រសិនបើមិនបានប្រើ aggregate functions។

ឧទាហរណ៍ 17: ស្វែងរកចំនួនបុគ្គលិក និងប្រាក់ខែសរុបនៅតាមសាខានីមួយៗ?

```
SELECT bno, COUNT(sno) AS count, SUM(salary) AS sum
FROM staff
GROUP BY bno
ORDER BY bno;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.17

bno	count	sum
B3	3	54000.00
B5	2	39000.00
B7	1	9000.00

(រូប 5.17)

SQL ដំណើរការ query នេះដូចខាងក្រោម:

1. SQL បែងចែកបុគ្គលិកជាក្រុមៗយោងទៅតាមលេខកូដសាខា (branch number) នីមួយៗ។ នៅក្រុមនីមួយៗ រាល់បុគ្គលិកទាំងអស់មានលេខកូដសាខាដូចគ្នា។ ចំពោះឧទាហរណ៍នេះយើងមាន 3 ក្រុម:

bno	sno	salary
B3	SG37	12000.00
B3	SG14	18000.00
B3	SG5	24000.00
B5	SL21	30000.00
B5	SL41	9000.00
B7	SA9	9000.00

COUNT(sno)	SUM(salary)
3	54000.00
2	39000.00
1	9000.00

2. SQL គណនាចំនួនបុគ្គលិកសរុប និងប្រាក់ខែសរុបតាមសាខានីមួយៗ។

3. ជាចុងបញ្ចប់ SQL តំរៀបលទ្ធផលតាមទំរង់លំដាប់កើនលើលេខកូដសាខា។

Restricting grouping (HAVING clause)

HAVING clause ត្រូវបានគេកសាងឡើងដើម្បីប្រើប្រាស់ជាមួយ GROUP BY clause ក្នុងការកំនត់ក្រុម (groups) បង្ហាញក្នុងលទ្ធផល ទោះបីជា HAVING និង WHERE មាន syntax ប្រហាក់ប្រហែលគ្នាតែវាត្រូវប្រាស់ក្នុងគោលបំណងខុសគ្នា។ WHERE clause ចំរាញ់ រឺកំនត់លើ individual rows រឺង HAVING clause ចំរាញ់ទៅលើ groups ។

ឧទាហរណ៍ 18: ស្វែងរកចំនួនបុគ្គលិក និងប្រាក់ខែសរុបនៅតាមសាខានីមួយៗចំពោះសាខាដែលមានបុគ្គលិកលើសពីម្នាក់?

```
SELECT bno, COUNT(sno) AS count, SUM(salary) AS sum
FROM staff
GROUP BY bno
HAVING COUNT(sno) > 1
ORDER BY bno;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.18

bno	count	sum
B3	3	54000.00
B5	2	39000.00

(រូប 5.18)

2. 5. Joining Tables

យើងដឹងហើយថា relational database ផ្ទុកនូវ tables ជាច្រើន ហើយ tables នីមួយៗផ្ទុកព័ត៌មានចំពោះប្រធានបទតែមួយប៉ុណ្ណោះ តែ tables ទាំងនោះមានទំនាក់ទំនងគ្នា។ ក្នុងករណីមួយចំនួន សំនួរខ្លះពាក់ព័ន្ធដល់ tables 2 រឺច្រើនដែលទាមទារអោយមានការតភ្ជាប់ tables សំរាប់ដំណើរការទិន្នន័យ។ ការតភ្ជាប់ tables ចែកជា Simple join, Inner Join, Left Join, Right Join និង Full Join ។

- **Simple Join**

```
SELECT
FROM Table1, Table2
WHERE Table1.Field = Table2.Field
```

ដើម្បីដំណើរការ simple join ទាមទារអោយធ្វើការបញ្ជូលឈ្មោះ tables លើសពី 1 ទៅកាន់ FROM clause ដែលខំណុដ្តាច់ដោយសញ្ញា comma និងធ្វើការកំនត់លក្ខខណ្ឌទៅលើ WHERE clause ។

ឧទាហរណ៍ 19: បង្ហាញឈ្មោះ renters ទាំងអស់ដែលបានទៅមើល property?

```
SELECT r.rno, fname, lname, pno, comments
FROM renter r, viewing v
WHERE r.rno = v.rno;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.19

rno	fname	lname	pno	comment
CR56	Aline	Stewart	PG36	
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR62	Mary	Tregear	PA14	no dining room
CR76	John	Kay	PG4	too remote

(រូប 5.19)

- **INNER Join**

SELECT
FROM Table1 INNER JOIN Table2 ON Table1.Field = Table2.Field

Inner Join ធ្វើការទាញទិន្នន័យចេញពី tables ទាំង 2 ដោយយកតែ records ណាដែលមានតំលៃនៅលើ field តភ្ជាប់ស្មើគ្នា។

ឧទាហរណ៍ 19: បង្ហាញឈ្មោះ renters ទាំងអស់ដែលបានទៅមើល property?

```
SELECT r.rno, fname, lname, pno, comments
FROM renter INNER JOIN viewing ON renter.rno = viewing.rno;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.19 ខាងលើ

- **LEFT OUTER Join**

SELECT
FROM Table1 LEFT OUTER JOIN Table2 ON Table1.Field = Table2.Field

Left Join ធ្វើការទាញទិន្នន័យទាំងអស់ចេញពី table ទី 1 និងទាញយកទិន្នន័យចេញពី table ទី 2 ដោយយកតែ records ណាមានតំលៃនៅលើ field តភ្ជាប់ស្មើគ្នា។

ឧទាហរណ៍ 20: បង្ហាញឈ្មោះ renters ណាដែលមិនបានទៅមើល property?

```
SELECT r.rno, fname, lname
FROM renter LEFT OUTER JOIN viewing ON renter.rno = viewing.rno;
```

លទ្ធផលបង្ហាញក្នុងរូប 5.20

rno	fname	Lname
CR74	Mike	Ritchie

(រូប 5.20)

- **RIGHT OUTER Join**

SELECT
FROM Table1 RIGHT OUTER JOIN Table2 ON Table1.Field = Table2.Field

Right Join ធ្វើការទាញទិន្នន័យទាំងអស់ចេញពី table ទី 2 និងទាញយកទិន្នន័យចេញពី table ទី 1 ដោយយកតែ records ណាមានតំលៃនៅលើ field តភ្ជាប់ស្មើគ្នា។

- **FULL Join**

SELECT
FROM Table1 FULL OUTER JOIN Table2 ON Table1.Field = Table2.Field
Full Join = Left Join + Right Join

2. 6. Database Updates

បញ្ជាសំរាប់ធ្វើការកែប្រែទិន្នន័យរក្សាក្នុង database ចែកជា:

- INSERT សំរាប់បញ្ចូលទិន្នន័យទៅកាន់ table
- UPDATE សំរាប់កែប្រែទិន្នន័យក្នុង table
- DELETE សំរាប់លប់ទិន្នន័យក្នុង table

Adding data to the database (INSERT)

```
INSERT INTO table_name[(column_list)]
VALUES (data_value_list)
```

បញ្ជា INSERT ជាបញ្ជាមួយសំរាប់ធ្វើការបញ្ចូលទិន្នន័យទៅកាន់ tables នៅក្នុង database។ table_name អាចជាឈ្មោះ table រឺ view ឯ column_list បង្ហាញពីបណ្តុំឈ្មោះ columns មួយ រឺច្រើនដែលខំណូផ្តាច់គ្នាដោយ comma និង data_value_list គឺជាតំលៃសំរាប់បញ្ចូល។ ប្រសិនបើយើងមិនបញ្ចូល column_list នោះវានឹងបញ្ចូលទិន្នន័យទៅតាមលំដាប់ដូចពេលដែលធ្វើការបង្កើត table ប៉ុន្តែបើយើងមិនបានបញ្ចូលតំលៃទៅលើ field ណាមួយ field នោះអាចទទួលយកតំលៃ DEFAULT រឺតំលៃ NULL បើគ្មាន DEFAULT។

ឧទាហរណ៍ 21: បញ្ចូល record ថ្មីទៅកាន់ Staff table ដោយផ្តល់ទិន្នន័យដូចខាងក្រោម

INSERT INTO staff
VALUES (‘SG16’, ‘Alan’, ‘Brown’,
’67 Endrick Rd, Glasgrow G32 8QR’, ‘014-211-3001’, ‘Assistant’,
‘M’, ‘05/25/1957’, 8300, ‘WN848391H’, ‘B3’)

Modifying data in the database (UPDATE)

```
UPDATE table_name
SET column_name1 = data_value1 [,column_name2 = data_value2...]
[WHERE search_condition]
```

បញ្ជា UPDATE ជាបញ្ជាមួយសំរាប់ធ្វើការកែប្រែទិន្នន័យក្នុង tables ហើយ SET ជាទីកន្លែងសំរាប់ធ្វើការកំនត់តំលៃថ្មី ប្រសិនបើគ្មាន WHERE វាធ្វើការកែប្រែទិន្នន័យក្នុង table ទាំងអស់។

ឧទាហរណ៍ 22: តំឡើងប្រាក់ខែ 3% អោយបុគ្គលិកទាំងអស់?

```
UPDATE staff
SET salary = salary * 1.03;
```

ឧទាហរណ៍ 23: តំឡើងប្រាក់ខែ 5% អោយបុគ្គលិកមានតួនាទីជា manager?

```
UPDATE staff
SET salary = salary * 1.05
WHERE position = 'Manager';
```

ឧទាហរណ៍ 24: តំឡើង David Ford (Sno = 'SG14') ទៅជា manager និងកែប្រែប្រាក់ខែទៅជា 18000 វិញ?

```
UPDATE staff
SET position = 'Manager', salary = 18000
WHERE sno = 'SG14';
```

Deleting data from the database (DELETE)

```
DELETE FROM table_name
[WHERE search_condition]
```

បញ្ជា DELETE ជាបញ្ជាមួយសំរាប់លុបទិន្នន័យក្នុង tables ប្រសិនបើគ្មាន WHERE វាធ្វើការលុបទិន្នន័យក្នុង table ទាំងអស់។

ឧទាហរណ៍ 25: លុបទិន្នន័យទាំងអស់ចេញពី viewing table

```
DELETE FROM viewing;
```

ឧទាហរណ៍ 26: លុបទិន្នន័យចេញពី viewing table ដែលទាក់ទង property PG4

```
DELETE FROM viewing
WHERE pno = 'PG4';
```

3. Data Definition Language

Data Definition Language (DDL) គឺជាបញ្ជាបង្កើត, កែប្រែ និងលុប database objects ដូចជា tables, domains, fields, views និង indexes។ ជាទូទៅ បញ្ជា DDL ចាប់ផ្តើមដោយពាក្យ CREATE, ALTER, DROP... ហើយជាជំហ្វងចូរសង្កេតមើលទៅលើប្រភេទទិន្នន័យជាមុនសិន។

3. 1. SQL Data Type

នៅពេលយើងបង្កើត field យើងត្រូវតែកំនត់ប្រភេទទិន្នន័យទៅអោយ field នោះ ហើយប្រភេទទិន្នន័យចែកចេញជាច្រើនប្រភេទដូចខាងក្រោម:

Character

Character សំរាប់ផ្ទុកប្រភេទទិន្នន័យជាតួអក្សរ, លេខដែលមិនទាក់ពាក់ព័ន្ធការគណនា ហើយចែកជា 2 ប្រភេទគឺ Variable-length និង fixed-length។ Fixed-length character មានលក្ខណៈកំនត់ជាក់លាក់ ឧទាហរណ៍ បើសិនជាយើងកំនត់ Name field មានប្រភេទទិន្នន័យជា fixed-length character ផ្ទុក 50 តួអក្សរ ប៉ុន្តែយើងប្រើប្រាស់អស់ 30 តួអក្សរនោះ ទំហំរក្សានៅលើ disk ត្រូវការ 50 តួអក្សរដោយយក 30 តួអក្សរដែលបញ្ចូល បន្ថែមជាមួយ space ចំនួន 20 ទៀតនៅខាងក្រោយ ខុសពី variable-length character ប្រសិនបើយើងកំនត់ 50 តួអក្សរដូចគ្នា ហើយប្រើប្រាស់អស់ 30 តួអក្សរនោះ space នៅលើ disk វាត្រូវការតែ 30 តួអក្សរតែប៉ុណ្ណោះ។

CHAR (length)
VARCHAR (length)

Bit data

Bit data ប្រើប្រាស់ទៅលើ field ណាដែលមានតំលៃតែ 2 គត់គឺ ពិត (True) និង មិនពិត (False) ដែលតំណាងដោយលេខ 0 រឺ 1 ។

BIT

Exact numeric data

Exact numeric data ប្រើប្រាស់ចំពោះ field ណាដែលផ្ទុកទិន្នន័យជាលេខសំរាប់គណនា ហើយត្រូវការការបង្ហាញជាក់លាក់។ Exact number data ផ្ទុកនូវ precision និង scale ដែល precision បង្ហាញពីចំនួនលេខទាំងអស់ដោយមិនគិតសញ្ញាទសភាគ ហើយ scale បង្ហាញពីចំនួនលេខខាងក្រោយក្បៀស។

NUMERIC (precision, scale)
DECIMAL (precision, scale)
INT
SMALLINT

Approximate numeric data

Approximate numeric data ប្រើប្រាស់ចំពោះ field ណាដែលផ្ទុកទិន្នន័យជាលេខសំរាប់គណនា ហើយការបង្ហាញពុំមានភាពជាក់លាក់។

FLOAT
REAL

Datetime data

Datetime data ប្រើប្រាស់ចំពោះ field ណាដែលផ្ទុកទិន្នន័យជាកាលបរិច្ឆេទ។

DATETIME
SMALLDATETIME

3. 2. Creating a Table (CREATE TABLE)

CREATE TABLE ជាបញ្ជាសំរាប់ធ្វើការបង្កើត table សំរាប់រក្សាទុកក្នុង database ។

```
CREATE TABLE table_name
    {(column_name data_type [NOT NULL] [UNIQUE]
    [DEFAULT defalut_value] [CHECK (search_condition)][,...])}
    [PRIMARY KEY (list_of_columns),]
    {[FOREIGN KEY (foreign_key_columns) REFERENCES
    parent_table_name (primary_key_columns)] [ON UPDATE
    CASCADE] [ON DELETE CASCADE] [,...]}
```

សមាសធាតុជាមូលដ្ឋាននៃ CREATE TABLE statement **គឺ** table_name, column_name, data_type **និង** NOT NULL **ប្រសិនបើយើងកំនត់** NOT NULL **ទៅលើ** field **ណា** field **នោះដាច់ខាតត្រូវតែបញ្ចូលទិន្នន័យជាដាច់ខាត បើមិនបញ្ចូលទេនឹង** error ។

DEFAULT **ប្រើសំរាប់តំលៃទុក ប្រសិនបើមិនបញ្ចូលទិន្នន័យ** field **នោះនឹងទទួលតំលៃ** default **នោះ ឧទាហរណ៍: កំនត់តំលៃ** default **ទៅលើ** field Sex **អោយទទួលតួអក្សរ** ‘M’ **នៅពេលមិនបញ្ចូលទិន្នន័យ។**

```
Sex Char(1) DEFAULT ‘M’
```

CHECK **ប្រើសំរាប់កំនត់តំលៃណាដែលអាចបញ្ចូលទៅកាន់** field **ឧទាហរណ៍: ចំពោះយើងបញ្ចូលទិន្នន័យបានតែ** 1 **តួអក្សរ ជាតួអក្សរ** ‘M’ **រឺ** ‘F’ ។

```
Sex Char(1) DEFAULT ‘M’ CHECK (VALUE IN(‘M’,’F’))
```

PRIMARY **សំរាប់កំនត់** field **ណាមានតួនាទីជា** primary key ។

```
PRIMARY KEY (sno)
```

FOREIGN KEY **សំរាប់ធ្វើការកំនត់ថា ទិន្នន័យបញ្ចូលលើ** field **តភ្ជាប់ រឺ** foreign key **ខាងផ្នែក** table many **អាចបញ្ចូលបានលុះត្រាតែទិន្នន័យនោះមាន រឺបង្ហាញក្នុង** field **តភ្ជាប់ រឺ** primary key **ខាងផ្នែក** table one ។

```
FOREIGN KEY (sno) REFERENCES staff (sno)
```

ON UPDATE CASCADE **ប្រើប្រាស់សំរាប់ដាក់កំរិតថា បើយើងធ្វើការកែប្រែទិន្នន័យលើ** primary key field **ខាងផ្នែក** table one **នោះតំលៃនៅលើ** field **តភ្ជាប់ខាង** table many **មានតំលៃស្មើគ្នាក៏កែប្រែតាមដែរ។**

ON DELETE CASCADE **ប្រើប្រាស់សំរាប់ដាក់កំរិតថា បើយើងធ្វើការលុបទិន្នន័យលើ** primary key field **ខាងផ្នែក** table one **នោះតំលៃនៅលើ** field **តភ្ជាប់ខាង** table many **មានតំលៃស្មើគ្នាក៏លុបតាមដែរ។**

ឧទាហរណ៍ 27: សរសេរឃ្លាបញ្ជាដើម្បីបង្កើត tables ខាងក្រោមនេះ

Branch (Bno, Street, Area, City, Pcode, Tel_No, Fax_No)
Staff (Sno, FName, LName, Address, Tel_No, Position, Sex, DOB, Salary, NIN, Bno)
Property_for_Rent (Pno, Street, Area, City, Pcode, Type, Rooms, Rent, Ono, Sno, Bno)
Renter (Rno, FName, LName, Address, Tel_No, Pref_Type, Max_Rent, Bno)
Owner (Ono, FName, LName, Address, Tel_No)
Viewing (Rno, Pno, Date, Comment)

```
CREATE TABLE branch(
        bno             VARCHAR(3)          NOT NULL,
        street          VARCHAR(25)         NOT NULL,
        area            VARCHAR(15),
        city            VARCHAR(15),
        pcode           VARCHAR(8),
        tel_no          VARCHAR(13),
        fax_no          VARCHAR(13),
        PRIMARY KEY (bno))

CREATE TABLE owner(
        ono             VARCHAR(5)          NOT NULL,
        fname           VARCHAR(15)         NOT NULL,
        lname           VARCHAR(15)         NOT NULL,
        address         VARCHAR(50),
        tel_no          VARCHAR(13),
        PRIMARY KEY (ono))

CREATE TABLE staff(
        sno             VARCHAR(5)          NOT NULL,
        fname           VARCHAR(15)         NOT NULL,
        lname           VARCHAR(15)         NOT NULL,
        address         VARCHAR(50),
        tel_no          VARCHAR(13),
        position        VARCHAR(10)         NOT NULL,
        sex             CHAR(1)             CHECK(VALUE IN('M','F')),
        dob             DATETIME,
        salary          DECIMAL(7,2)        NOT NULL,
        nin             CHAR(9),
        bno             VARCHAR(3)          NOT NULL,
        PRIMARY KEY (sno),
        FOREIGN KEY (bno) REFERENCES branch (bno))

CREATE TABLE renter(
        rno             VARCHAR(5)          NOT NULL,
        fname           VARCHAR(15)         NOT NULL,
        lname           VARCHAR(15)         NOT NULL,
        address         VARCHAR(50),
        tel_no          VARCHAR(13),
        pref_type       VARCHAR(15),
        max_rent        INT,
```

```
        bno             VARCHAR(3)          NOT NULL,
        PRIMARY KEY (rno),
        FOREIGN KEY (bno) REFERENCES branch (bno))
    CREATE TABLE property_for_rent(
        pno             VARCHAR(5)          NOT NULL,
        street          VARCHAR(25)         NOT NULL,
        area            VARCHAR(15),
        city            VARCHAR(15)         NOT NULL,
        pcode           VARCHAR(8),
        type            CHAR(1)             NOT NULL,
        rooms           SMALLINT            NOT NULL,
        rent            DECIMAL(6,2)        NOT NULL,
        ono             VARCHAR(5)          NOT NULL,
        sno             VARCHAR(5),
        bno             VARCHAR(3)          NOT NULL,
        PRIMARY KEY (pno),
        FOREIGN KEY (ono) REFERENCES owner (ono)
        FOREIGN KEY (sno) REFERENCES staff (sno),
        FOREIGN KEY (bno) REFERENCES branch (bno))
    CREATE TABLE viewing(
        rno             VARCHAR(5)          NOT NULL,
        pno             VARCHAR(5)          NOT NULL,
        date            DATETIME,
        comment         VARCHAR(50),
        PRIMARY KEY (rno, pno),
        FOREIGN KEY (rno) REFERENCES renter (rno)
        FOREIGN KEY (pno) REFERENCES property_for_rent (pno))
```

3. 3. Removing a Table (DROP TABLE)

DROP TABLE គឺជាបញ្ជាសំរាប់លុប table ចេញពី database។

```
DROP TABLE table_name
```

ឧទាហរណ៍ 28: សរសេរឃ្លាបញ្ជាដើម្បីលុប staff table

```
DROP TABLE staff
```

3. 4. Altering a Table (ALTER TABLE)

ALTER TABLE គឺជាបញ្ជាសំរាប់កែប្រែរចនាសម្ពន្ធនៃ table ដោយអាចបន្ថែម រឺលុប field ចេញពី table រឺក៏កែប្រែ field property។

```
ALTER TABLE table_name
  [ADD column_name data_type [NOT NULL] [UNIQUE]
  [DEFAULT default_option] [CHECK (search_condition)]]
  [DROP column_name]
```

ឧទាហរណ៍ 29: សរសេរឃ្លាបញ្ជាដើម្បីបន្ថែម Sex field ទៅកាន់ renter table

```
ALTER TABLE renter
ADD sex char(1) NOT NULL
```

ឧទាហរណ៍ 30: សរសេរឃ្លាបញ្ជាដើម្បីលុប Sex field ចេញពី renter table

```
ALTER TABLE renter
DROP sex
```

3. 5. Creating an Index (CREATE INDEX)

CREATE INDEX គឺជាបញ្ជាសំរាប់បង្កើត index ទៅលើ field នៃ table ណាមួយ។

```
CREATE INDEX index_name ON
table_name (column [ASC|DESC] [,...])
```

ឧទាហរណ៍ 31: សរសេរឃ្លាបញ្ជាដើម្បីបង្កើត index ទៅលើ field nin

```
CREATE INDEX nin_ind ON staff (nin)
```

3. 6. Removing an Index (DROP INDEX)

DROP INDEX គឺជាបញ្ជាសំរាប់លុប index ចេញពី table។

```
DROP INDEX index_name
```

ឧទាហរណ៍ 28: សរសេរឃ្លាបញ្ជាដើម្បីលុប index ឈ្មោះ nin_ind

```
DROP INDEX nin_ind
```

Review Questions

1. អ្វីទៅជា SQL? តើវាចែកជាប៉ុន្មានប្រភេទ?
2. ចូរពន្យល់ពី clause នីមួយៗនៅក្នុងបញ្ជា SQL?
3. ចូរពន្យល់ពីភាពខុសគ្នារវាង WHERE clause និង HAVING clause ព្រមទាំងលើកឧទាហរណ៍មកបញ្ជាក់ផង?
4. ពន្យល់ពីដំណើរការរបស់ GROUP BY?