Chapter 3 : Structured Query Language (SQL)

Introduction
Writing SQL Command
Literals
SQL-DML
Aggregate Functions
SQL-DDL
Query-By-Example (QBE)
Integrity Enhancement Features (IEF)

Introduction

- stands for Structured Query Language.
- ideally, database language should allow user to:
  - create the database and relation structures;
  - perform insertion, modification, deletion of data from relations;
  - perform simple and complex queries.
- used in 2 ways :-
  a) embedded – by allowing SQL statements to be coded into host language such as JAVA, COBOL, C++ or FORTRAN.
  b) interactive (dynamic SQL) – by entering statements at a terminal.

- consists of 2 main components :-
  a) DDL – for defining the database structure.
  b) DML – for retrieving, inserting, deleting and updating data.
- SQL is relatively easy to learn:
  a) it is non-procedural - you specify *what* information you require, rather than *how* to get it;
  b) it is essentially free-format.
- consists of standard English words:
  a) CREATE TABLE Staff(staffNo VARCHAR(5),
  lName VARCHAR(15),
  salary DECIMAL(7,2));
  b) INSERT INTO Staff
  VALUES('SG16', 'Brown', 8300);

c) SELECT staffNo, lName, salary
FROM Staff
WHERE salary > 10000;

- can be used by range of users including DBAs, management, application developers and other types of end users.
- an ISO standard now exists for SQL, making it both the formal and *de facto* standard language for relational databases.
- it must be portable.

Writing SQL Command

- consists of **reserved words** and **user-defined words**.
- ***reserved words*** - a fixed part of SQL and must be spelt exactly as required and cannot be split across lines.
- ***user-defined words*** - made up by user and represent names of various database objects such as relations, columns, views.
- most components of an SQL statement are *case insensitive*, except for **literal** character data**.**
  - literals are constants used in SQL statements.
  - all non-numeric literals must be enclosed in single quotes (e.g. 'London').
  - all numeric literals must not be enclosed in quotes (e.g. 650.00).

- requires the use of **statement terminator (;)** to mark the end of each SQL statement.
- more readable with **indentation** and **lineation**:
  - each clause should begin on a new line.
  - start of a clause should line up with start of other clauses.
  - if clause has several parts, should each appear on a separate line and be indented under start of clause.
- use ***Backus Naur Form*** (BNF):
  a) *upper case letter* - **reserved words**.
  b) *lower case letter* – **user-defined words**.
  c) *vertical bar* (|) - **choice among alternatives**. a | b | c
  d) *curly braces* { } – **required element**. {a}
  e) *square bracket* [ ] – **optional element**. [a]
  f) *ellipsis* (…) – **optional repetition of an item zero or more times.**

SQL-DML

SQL DML statements :-

SELECT – to retrieve & display data from one or more database tables.

INSERT – to insert new data into a table.

UPDATE – to update data in a table.

DELETE – to delete data from a table.

general statement for SQL :-

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

-forms groups of rows with the same column value

-specifies the order of the output.

eliminate duplicates

column_name @ statement

specifies the table or tables to be used.

to name a column

table name @ view you have access to

optional abbreviation for table name

filters the groups subject to some condition

filters the rows subject to some condition.

Example 3.1 All Columns, All Rows

List full details of all staff.

```
SELECT staffNo, fName, lName, address,
       position, sex, DOB, salary, branchNo
FROM Staff;
```

Can use * as an abbreviation for 'all columns':

```
SELECT *
FROM Staff;
```

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

Example 3.2 Specific Columns, All Rows

Produce a list of salaries for all staff, showing only staff number, first and last names, and salary.

```
SELECT staffNo, fName, lName, salary
FROM Staff;
```

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG37	Ann	Beech	12000.00
SG14	David	Ford	18000.00
SA9	Mary	Howe	9000.00
SG5	Susan	Brand	24000.00
SL41	Julie	Lee	9000.00

Example 3.3 Use of DISTINCT

List the property numbers of all properties that have been viewed.

```
SELECT propertyNo
FROM Viewing;
```

propertyNo
PA14
PG4
PG4
PA14
PG36

Use DISTINCT to eliminate duplicates:

```
SELECT DISTINCT propertyNo
FROM Viewing;
```

propertyNo
PA14
PG4
PG36

Example 3.4 Calculated Fields

Produce list of monthly salaries for all staff, showing staff number, first/last name, and salary.

```
SELECT staffNo, fName,
       lName, salary/12
FROM Staff;
```

staffNo	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

To name column, use **AS clause**:

```
SELECT staffNo, fName, lName, salary/12
       AS monthlySalary
FROM Staff;
```


Tutorial

Contractor (contractorNo, contractorName, telNo, status, state)
Project (projectNo, projectName, startDate, finishDate, state)
Contractorproject (contractorNo, projectNo, cost)

- List all project details.
- List contractor number, name and telephone number.

STAF

Emp_Num	Emp_LName	Emp_FName	Emp_Salary	Emp_HireDate	Job_Code
101	News	John	20000	11/8/92	505
102	Senior	David	30000	7/12/87	501
103	Arbough	June	15000	12/1/94	500
104	Ramoras	Anne	35000	11/15/85	501
105	Johnson	Alice	27000	2/1/91	502
106	Smithfield	William	18000	6/23/90	500
107	Alonzo	Maria	15000	10/10/91	500
108	Washington	Ralph	37000	8/22/89	501
109	Smith	Larry	39000	7/18/95	501

- List the staff that has no duplicate Job_Code.

WHERE clause

- 4 basic search conditions :-

a) ***comparison***
- compare the value of one expression to the value of another expression.

b) ***range*** **- BETWEEN/NOT BETWEEN**
- test whether the value of an expression falls within a specified range of values.

c) ***set membership*** **- IN/NOT IN**
- test whether the value of an expression equals one of a set of values.

d) ***pattern match*** **- LIKE/NOT LIKE**
- test whether a string matches a specified pattern.

Example 3.5 Comparison Search Condition

List all staff with a salary greater than 10,000.

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > 10000;
```

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Assistant	12000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

Example 3.6 Compound Comparison Search Condition

List addresses of all branch offices in London or Glasgow.

```
SELECT *
FROM Branch
WHERE city = 'London' OR city = 'Glasgow';
```

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B003	163 Main St	Glasgow	G11 9QX
B002	56 Clover Dr	London	NW10 6EU

Example 3.7 Range Search Condition

List all staff with a salary between 20,000 and 30,000.

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary BETWEEN 20000 AND 30000;
```

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG5	Susan	Brand	Manager	24000.00

Could also write:

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary >=20000 AND salary <= 30000;
```


Example 3.8 Set Membership

List all managers and supervisors.

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position IN ('Manager', 'Supervisor');
```

could have expressed this as:

staffNo	fName	lName	position
SL21	John	White	Manager
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position='Manager' OR
      position='Supervisor';
```

IN is more efficient when set contains many values.

Example 3.9 Pattern Matching

Find all owners with the string 'Glasgow' in their address.

```
SELECT ownerNo, fName, lName, address, telNo
FROM PrivateOwner
WHERE address LIKE '%Glasgow%';
```

ownerNo	fName	lName	address	telNo
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

- SQL has two special pattern matching symbols:
  - %, *: sequence of zero or more characters;
  - _ (underscore): any single character.
- LIKE '%Glasgow%' means a sequence of characters of any length containing '*Glasgow*'.
- Example
  - *address* ***LIKE 'H%'***
  - -1st character starts with H & but the rest of the string can be anything.
  - *address* ***LIKE 'H_ _ _'***
  - there are 4 characters in the string, 1st must be an H.
  - *address* ***LIKE '%e'***
  - any sequences of characters of length at least 1, with the last character an e.

Example 3.10 Single Column Ordering

- **Sorting results** (**ORDER BY** clause)
  - allows to order the retrieved records in ascending (ASC) or descending (DESC).
- List salaries for all staff, arranged in descending order of salary.

```
SELECT staffNo, fName, lName, salary
FROM Staff
ORDER BY salary DESC;
```

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG5	Susan	Brand	24000.00
SG14	David	Ford	18000.00
SG37	Ann	Beech	12000.00
SA9	Mary	Howe	9000.00
SL41	Julie	Lee	9000.00

Tutorial

Contractor (<u>contractorNo</u>, contractorName, telNo, status, state)
Project (<u>projectNo</u>, projectName, startDate, finishDate, state)
Contractorproject (<u>contractorNo</u>, <u>projectNo</u>, cost)

- List all contractor details for contractor from 'Kuala Lumpur'.
- List contractorNo, projectNo, cost for contractor that get a project with cost more than RM50,000 and less than RM100,000.
- List the contractorNo that handles a projectNo 'P101' alphabetically ordered by contractorNo.

STAF

Emp_Num	Emp_LName	Emp_FName	Emp_Salary	Emp_HireDate	Job_Code
101	News	John	20000	11/8/92	505
102	Senior	David	30000	7/12/87	501
103	Arbough	June	15000	12/1/94	500
104	Ramoras	Anne	35000	11/15/85	501
105	Johnson	Alice	27000	2/1/91	502
106	Smithfield	William	18000	6/23/90	500
107	Alonzo	Maria	15000	10/10/91	500
108	Washington	Ralph	37000	8/22/89	501
109	Smith	Larry	39000	7/18/95	501

- List all the attributes for Job_Code 502.
- List Emp_Num, Emp_Salary for all staff with salary between 15000 and 27000.
- List details of Emp_Num, Emp_HireDate arranged in ascending order of Emp_HireDate.

Aggregate Functions

- ISO standard defines 5 aggregate functions :-
  a) **COUNT** - the no. of values in a specified column.
  b) **SUM** - the sum of the values in a specified column.
  c) **AVG** - the average of the values in a specified column.
  d) **MIN** - the smallest value in a specified column.
  e) **MAX** - the largest values in a specified column.
- COUNT, MIN and MAX apply to both numeric and non-numeric fields.
- SUM and AVG may be used in numeric fields only.
- each functions eliminates nulls first and operates only on remaining non-null values except for COUNT(*).

- can be used only in SELECT and HAVING clauses.
- COUNT(*) counts all rows of a table, regardless of whether nulls or duplicate values occur.
- Can use DISTINCT before column name to eliminate duplicates.
- DISTINCT has no effect with MIN/MAX, but may have with SUM/AVG.
- If SELECT list includes an aggregate function and there is no GROUP BY clause, SELECT list cannot reference a column out with an aggregate function. For example, the following is illegal:

```
SELECT staffNo, COUNT(salary)
FROM Staff;
```

Example 3.11 Use of COUNT(*)

- How many properties cost more than 350 per month to rent?

```
SELECT COUNT(*) AS myCount
FROM PropertyForRent
WHERE rent > 350;
```

myCount
5

Example 3.12 Use of COUNT(DISTINCT)

- How many different properties viewed in May '04?

```
SELECT COUNT(DISTINCT propertyNo)
    AS myCount
FROM Viewing
WHERE viewDate BETWEEN '1-May-04'
    AND '31-May-04';
```

myCount
2

Example 3.13 Use of COUNT and SUM

- Find number of Managers and sum of their salaries.

```
SELECT COUNT(staffNo) AS myCount,
       SUM(salary) AS mySum
FROM Staff
WHERE position = 'Manager';
```

myCount	mySum
2	54000.00

Example 3.14 Use of MIN, MAX, AVG

- Find minimum, maximum and average staff salary.

```
SELECT MIN(salary) AS myMin,
    MAX(salary) AS myMax,
    AVG(salary) AS myAvg
FROM Staff;
```

myMin	myMax	myAvg
9000.00	30000.00	17000.00

SELECT Statement - Grouping

- Use GROUP BY clause to get sub-totals.
- SELECT and GROUP BY closely integrated: each item in SELECT list must be *single-valued per group*, and SELECT clause may only contain:
  - column names
  - aggregate functions
  - constants
  - expression involving combinations of the above.

Example 3.15 Use of GROUP BY

- Find number of staff in each branch and their total salaries.

```
SELECTbranchNo, COUNT(staffNo) AS myCount,
      SUM(salary) AS mySum
FROM Staff
GROUP BY branchNo
ORDER BY branchNo;
```

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

Restricted Groupings – HAVING clause

- HAVING clause is designed for use with GROUP BY to restrict groups that appear in final result table.
- Similar to WHERE, but WHERE filters individual rows whereas HAVING filters groups.
- Column names in HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function.

Example 3.16 Use of HAVING

For each branch with more than 1 member of staff, find number of staff in each branch and sum of their salaries.

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00

```
SELECT branchNo,
    COUNT(staffNo) AS myCount,
    SUM(salary) AS mySum
FROM Staff
GROUP BY branchNo
HAVING COUNT(staffNo) > 1
ORDER BY branchNo;
```


Tutorial

Contractor (contractorNo, contractorName, telNo, status, state)
Project (projectNo, projectName, startDate, finishDate, state)
Contractorproject (contractorNo, projectNo, cost)

- Find the total number of project handled by contractor_No 'A1234567'.

STAF

Emp_Num	Emp_LName	Emp_FName	Emp_Salary	Emp_HireDate	Job_Code
101	News	John	20000	11/8/92	505
102	Senior	David	30000	7/12/87	501
103	Arbough	June	15000	12/1/94	500
104	Ramoras	Anne	35000	11/15/85	501
105	Johnson	Alice	27000	2/1/91	502
106	Smithfield	William	18000	6/23/90	500
107	Alonzo	Maria	15000	10/10/91	500
108	Washington	Ralph	37000	8/22/89	501
109	Smith	Larry	39000	7/18/95	501

- Find the total number of staff with salary more than 25000.

Product

P_Code	P_Descript	P_Indate	P_Onhand	P_Price	V_Code
11QER/31	Powerpainter,15psi 3 nozzle	12/2/96	8	$109.99	25595
13-Q2/P2	7.25-in.pwr.saw blade	11/12/96	32	$14.99	21344
14-Q1/L3	9.00 in.pwr.saw blade	11/12/96	18	$17.49	21344
1546-QQ2	Hrd.cloth,1/2 in. 2X50	8/14/96	15	$39.95	23119
1558-QW1	Hrd.Cloth,1/2 in.3X50	8/14/96	23	$43.99	23119
2232/QTY	B&D jigsaw,12 in blade	10/25/96	8	$109.92	24288
2232/QWE	B&D Jigsaw, 8 in blade	9/23/96	6	$99.87	24288
2238/QPD	B&D Cordless drill,1/2 in	10/19/96	12	$98.95	25595
23109-HB	Claw Hammer	11/19/96	23	$9.95	21225
23114-AA	Sledge hammer	12/1/96	8	$14.40	
54778-2T	Rat Tail	6/14/96	43	$4.99	21344
89-WRE-Q	Hrnut chain	7/5/96	11	$256.99	24288

- Find the total number of P_Descript that cost more than $100.00
- Find the total number of product that cost more than $150 and sum the price.

Subqueries

- Some SQL statements can have a SELECT embedded within them.
- A subselect can be used in WHERE and HAVING clauses of an outer SELECT, where it is called a *subquery* or *nested query*.
- Subselects may also appear in INSERT, UPDATE and DELETE statements.

Example 3.17 Subquery with Equality

List staff who work in branch at '163 Main St'.

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo =
      (SELECT branchNo
       FROM Branch
       WHERE street = '163 Main St');
```

staffNo	fName	lName	position
SG37	Ann	Beech	Assistant
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

- Inner SELECT finds branch number for branch at '163 Main St' ('B003').
- Outer SELECT then retrieves details of all staff who work at this branch.
- Outer SELECT then becomes:

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo = 'B003';
```


Example 3.18 Nested subquery: use of IN

- List properties handled by staff at '163 Main St'.

```
SELECT propertyNo, street, city, postcode,
    type, rooms, rent
FROM PropertyForRent
WHERE staffNo IN
    (SELECT staffNo
     FROM Staff
     WHERE branchNo =
          (SELECT branchNo
           FROM Branch
           WHERE street = '163 Main St'));
```

propertyNo	street	city	postcode	type	rooms	rent
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375
PG21	18 Dale Rd	Glasgow	G12	House	5	600

Multi-Table Queries

- Can use subqueries provided result columns come from same table.
- If result columns come from more than one table must use a join.
- To perform join, include more than one table in FROM clause.
- Use comma as separator and typically include WHERE clause to specify join column(s).
- Also possible to use an alias for a table named in FROM clause.
- Alias is separated from table name with a space.
- Alias can be used to qualify column names when there is ambiguity.

Example 3.19 Simple Join

- List names of all clients who have viewed a property along with any comment supplied.

```
SELECT c.clientNo, fName, lName,
            propertyNo, comment
FROM Client c, Viewing v
WHERE c.clientNo = v.clientNo;
```

- Only those rows from both tables that have identical values in the clientNo columns (c.clientNo = v.clientNo) are included in result.

clientNo	fName	lName	propertyNo	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

Example 3.20 Three Table Join

- For each branch, list staff who manage properties, including city in which branch is located and properties they manage.

```
SELECT b.branchNo, b.city, s.staffNo, fName,
      lName,propertyNo
FROM Branch b, Staff s, PropertyForRent p
WHERE b.branchNo = s.branchNo AND
      s.staffNo = p.staffNo
ORDER BY b.branchNo, s.staffNo, propertyNo;
```

branchNo	city	staffNo	fName	lName	propertyNo
B003	Glasgow	SG14	David	Ford	PG16
B003	Glasgow	SG37	Ann	Beech	PG21
B003	Glasgow	SG37	Ann	Beech	PG36
B005	London	SL41	Julie	Lee	PL94
B007	Aberdeen	SA9	Mary	Howe	PA14

Tutorial

Contractor (contractorNo, contractorName, telNo, status, state)
Project (projectNo, projectName, startDate, finishDate, state)
Contractorproject (contractorNo, projectNo, cost)

- List full details of project handled by contractor from Kuala Lumpur.

STUDENT (icNo, name)
SUBJECT (subjectCode, subjectName)
STUDSUBJECT (icNo, subjectCode)
TEACHER (staffNo, name, position, telNo, subjectCode)

- List the teacher's detail that teach Database Concepts subject.
- List the student's name and the lecturer's name for JAVA subject.

INSERT

INSERT INTO TableName [(columnList)]
VALUES (dataValueList)

- Insert a new row into Staff table supplying data for all columns.

```
INSERT INTO Staff
VALUES ('SG16', 'Alan', 'Brown', 'Assistant', 'M',
  Date '1957-05-25', 8300, 'B003');
```


Example 3.21 INSERT using Defaults

- Insert a new row into Staff table supplying data for all mandatory columns.

```
INSERT INTO Staff (staffNo, fName, lName,
            position, salary, branchNo)
VALUES ('SG44', 'Anne', 'Jones',
            'Assistant', 8100, 'B003');
```

- Or

```
INSERT INTO Staff
VALUES ('SG44', 'Anne', 'Jones', 'Assistant',
            NULL, NULL, 8100, 'B003');
```


UPDATE

UPDATE TableName
SET columnName1 = dataValue1
[, columnName2 = dataValue2...]
[WHERE searchCondition]

- *TableName* can be name of a base table or an updatable view.
- SET clause specifies names of one or more columns that are to be updated.
- WHERE clause is optional:
  - if omitted, named columns are updated for all rows in table;
  - if specified, only those rows that satisfy *searchCondition* are updated.

Example 3.22/23 UPDATE All Rows

- Give all staff a 3% pay increase.

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

- Give all Managers a 5% pay increase.

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


Example 3.24 UPDATE Multiple Columns

- Promote David Ford (staffNo='SG14') to Manager and change his salary to 18,000.

```
UPDATE Staff
SET position = 'Manager', salary = 18000
WHERE staffNo = 'SG14';
```


DELETE

DELETE FROM TableName
[WHERE searchCondition]

- *TableName* can be named of a base table or an updatable view.
- *searchCondition* is optional; if omitted, all rows are deleted from table. This does not delete table. If *search_condition* is specified, only those rows that satisfy condition are deleted.

Example 3.25/26 DELETE Specific Rows

- Delete all viewings that relate to property PG4.

```
DELETE FROM Viewing
WHERE propertyNo = 'PG4';
```

- Delete all records from the Viewing table.

```
DELETE FROM Viewing;
```


SQL-DDL

- allows us to create and destroy database objects (schema, domains, tables, views and indexes).
- examples DDL :-

CREATE SCHEMA	**DROP SCHEMA**
CREATE TABLE	**DROP TABLE**
CREATE DOMAIN	**DROP DOMAIN**
CREATE VIEW	**DROP VIEW**
CREATE INDEX	**DROP INDEX**
ALTER DOMAIN	**ALTER TABLE**

CREATE TABLE

- Creates a table with one or more columns of the specified *dataType*.
- allows us to create the base table structures for the relations to be located in the database.
- With NOT NULL, system rejects any attempt to insert a null in the column.
- Can specify a DEFAULT value for the column.
- Primary keys should always be specified as NOT NULL.
- FOREIGN KEY clause specifies FK along with the referential action.

Create a table.

To illustrate the table creation process, we create the structures for the PropertyForRent table.

```
CREATE DOMAIN OwnerNumber AS VARCHAR(5)
  CHECK (VALUE IN (SELECT ownerNo
  FROM PrivateOwner));
CREATE DOMAIN StaffNumber AS VARCHAR(5)
  CHECK (VALUE IN (SELECT staffNo FROM Staff));
CREATE DOMAIN PNumber AS VARCHAR(5);
CREATE DOMAIN PRooms AS SMALLINT;
    CHECK(VALUE BETWEEN 1 AND 15);
CREATE DOMAIN PRent AS DECIMAL(6,2)
    CHECK(VALUE BETWEEN 0 AND 9999.99);
```



```
CREATE TABLE PropertyForRent (
  propertyNo PNumber NOT NULL,
  rooms     PRooms          NOT NULL DEFAULT 4,
  rent      PRent           NOT NULL,   DEFAULT 600,
  ownerNo   OwnerNumber  NOT NULL,
  staffNo       StaffNumber
      Constraint StaffNotHandlingTooMuch ….
  branchNo      BranchNumber        NOT NULL,
  PRIMARY KEY (propertyNo),
  FOREIGN KEY (staffNo) REFERENCES Staff
   ON DELETE SET NULL ON UPDATE CASCADE ….);
```


ALTER TABLE

- Add a new column to a table.
- Drop a column from a table.
- Add a new table constraint.
- Drop a table constraint.
- Set a default for a column.
- Drop a default for a column.

Example 3.27 - ALTER TABLE

- Change Staff table by removing default of 'Assistant' for position column and setting default for gender column to female ('F').

```
ALTER TABLE Staff
    ALTER position DROP DEFAULT;
ALTER TABLE Staff
    ALTER gender SET DEFAULT 'F';
```


- DROP TABLE (removing a table)
  - remove a table including its contents.
  - format :-
    DROP TABLE *table_name*
    [RESTRICT | CASCADE]

SQL automatically drops a dependent objects.

if there are any objects that depend on this table, SQL does not allow the DROP TABLE request to proceed.

- *Remove the Staff table.*

DROP TABLE *staff;*

Tutorial

MEMBER (memberID, name)
VIDEO (videoCode, title, category)
RENTAL (memberID, videoCode, quantity)
SUPPLIER (supplierNo, name, address, videoCode)

- create table **MEMBER** based on the relations given using **CREATE** statements.
- use **INSERT** statement to insert two rows of record into **MEMBER** relation. The records are:

row 1	**M101**	**Amin**
row 2	**M102**	**Nina**

- add an attribute named *director* to VIDEO relation.

QBE (Query By Example)

- graphical-based way of querying the database that provides facilities such as defining, manipulating, updating and controlling the relational database.
- the differences with SQL :-
  a) graphical interface – type query by providing the table to the system.
  b) productive way to interact with a terminal by non-technical end-user.
  c) for simple query and does not involve too many tables.

Integrity Enhancement Feature (IEF)

- facilities provided by the 1992 ISO SQL standard for integrity control.
- consists of constraints → to impose for protecting the database from becoming inconsistent.
- 5 types of integrity constraints :-
  - *required data*
  - *domain constraints*
  - *entity integrity*
  - *referential integrity*
  - *enterprise constraints.*
- can be defined in the CREATE and ALTER TABLE

Required Data

'some columns must contain a valid value; they are not allowed to contain missing values or nulls.'

- example :- *every member of staff must have an associated job position.*
- the ISO standard provides the NOT NULL column specifier in the CREATE and ALTER TABLE to provide this type of constraint.

position **VARCHAR**(9) NOT NULL

Domain Constraints

'every column has a domain, in other words a set of legal values.'

- example :- *the gender of a member of staff is either 'M' or 'F'.*
- provides 2 mechanisms in CREATE and ALTER TABLE:-
  - *CHECK clause*

- allows a constraint to be defined on a column or the whole table.

- format :-

CHECK *(search_condition)*

- example :-

gender **CHAR NOT NULL CHECK** (*gender* **IN** ('*M*', '*F*'))

CREATE DOMAIN statement

- allows domain to be defined more explicitly.

- format :-

CREATE DOMAIN *domain_name*
[**AS**] *data_type*
[**DEFAULT** *default_option*]
[**CHECK** (*search_condition*)]

- example :-

CREATE DOMAIN *gender_type* **AS CHAR**
CHECK (**VALUE IN** ('*M*', '*F*'));

- can be removed from the database by using **DROP DOMAIN**.

Entity Integrity

'***the primary key of a table must contain a unique, non-null value for each row***.'

- example :- *each row of the Staff table has a unique value for the staff number Sno.*

PRIMARY KEY(sno)

- with composite key :- **PRIMARY KEY**(rno, pno)

OR rno **CHAR(5) NOT NULL**,
pno **CHAR(5) NOT NULL**,
UNIQUE (rno, pno)

Referential Integrity

'***if the foreign key contains a value, that value must refer to an existing, valid row in the parent table.***'

- example :- *the branch number column Bno in the Staff table links the member of staff to that row in the Branch table where he or she works.*

FOREIGN KEY(bno) **REFERENCES** branch

Enterprise Constraints

'updates to tables may be constrained by enterprise rules governing the real-world transactions that are represented by the updates.'

- example :- *define rule that prevents a member of staff from managing more than 10 properties at the same time.*
- format :-

CREATE ASSERTION *assertion_name*
CHECK (*search_condition*)

- example :-

```
CREATE ASSERTION staff_not_handling_too_much
CHECK (NOT EXISTS
      (SELECT sno
       FROM property_for_rent
       GROUP BY sno
       HAVING COUNT(*) > 10));
```