```
5
>> Join Operations:
        >> Natural Join = Natural Inner Join:
                A natural join B
                A natural inner join B
                =
                A inner join B on
                A.id = B.id

        >> Outer Join: join then adds tuples from a relation that doesn't match
tuples in other relation & uses null values
                >> Left:
                        A natural left outer join B
                        ==
                        A left outer join B on
                        A.id = B.id
                >> Right:
                        A natural right outer join B
                        =
                        A right outer join B on
                        A.id = B.id
                >> Full:
                        A natural full outer join B
                        =
                        A full outer join B on
                        A.id = B.id

        >> Join Conditions:
                natural
                on <predicate> --> condition (like where clause)
                using (Mutual Column)

>> View: provides a mechanism to hide certain data from the view of certain users
(virtual relations)\
        create view faculty as
        select ID, name, dept_name
        from instructor

        select name
        from faculty
        where dept_name = ‘Biology’

        # You can create a view from another view

        # You can update views only under these conditions (and the table will be
updated too):
                The from clause has only one database relation.
                The select clause contains only attribute names of the relation,
and does not have any expressions, aggregates, or distinct specification.
                Any attribute not listed in the select clause can be set to null
```

The query does not have a group by or having clause.

>> Materialized view: if the actual relations change, the view is kept up-to-date.
by these ways:
View maintenance can be done immediately when any of the relations on which the view is defined is updated.
View maintenance can be performed lazily, when the view is accessed.
Some systems update materialized views only periodically

>> Transaction: a sequence of queries and/or update statements which are Atomic
>> begin implicitly and ended by one of the following:
* Commit work - commits the current transaction then a new will start
* Rollback work - causes the current transaction to be rolled back and undoes updates (restore)

>> Integrity Constraints: guard against accidental damage
>> Not Null
>> Unique: states that attributes form a candidate key
>> Check (a Predicate)
>> Referential Integrity: ensure that a value appears in a relation for a set of attributes also appears for a certain set of attributes in another relation

```
create table section (
        course_id varchar (8),
        sec_id varchar (8),
        semester varchar (6) not null, check (semester in (’Fall’, ’Winter’, ’Spring’, ’Summer’),
        year numeric (4,0),
        building varchar (15),
        room_number varchar (7),
        time slot id varchar (4),
        primary key (course_id, sec_id, semester, year),
        foreign key (course_id) references course,

        check (year > 1759 and year < 2100)
);
```

>> Cascading Actions: delete/update on the referenced relation cascades on the referencing relation
on delete/update: cascade, set null, set default

```
create table course (
        …
        dept_name varchar(20),
        foreign key (dept_name) references department

                on delete cascade
                on update cascade,
        …
)
```

```
        >> Complex Check Clauses:
                # subquery in check clause not supported
                create assertion credits_earned_constraint check
                (not exists (select ID
                                         from student
                                         where tot_cred <> (select sum(credits)

from takes natural join course

where student.ID= takes.ID and

grade is not null and grade<> ’F’)
                                        )
                )
>> Built-in Data Types:
        date ‘2005-7-27’
        time ‘09:00:30’
        timestamp  ‘2005-7-27 09:00:30.75’
        interval:  period of time

>> Default Values:
        create table student
                (ID varchar (5),
                name varchar (20) not null,
                dept_name varchar (20),
                tot_cred numeric (3,0) default 0,
                primary key (ID)
        )

        insert into student(ID,name,dept_name)
                values(’12789’, ’Newman’, ’Comp. Sci.’);

>> Index Creation:
        create table student
                (ID varchar (5),
                name varchar (20) not null,
                dept_name varchar (20),
                tot_cred numeric (3,0) default 0,
                primary key (ID)
        )

        create index studentID_index on student(ID)

>> Large-Object Types: Large objects (photos, videos, CAD files, etc.) are stored
as a large object
        blob: binary large object
        clob: character large object

        B clob(10KB)
        A blob(10MB)
        # When a query returns a large object, a “locator” is returned that can use
to fetch the large object in small pieces, rather than all at once

>> User-Defined Types: two forms --> 1. distinct types 2. structured data types

        create type Dollars as numeric (12,2) final

        create table department
                (dept_name varchar (20),
                building varchar (15),
                budget Dollars
        );

        # Values can be cast to another domain:
                cast (department.budget  to numeric (12,2))

>> Domains:
        # Types and domains are similar. Domains can have constraints

        create domain person_name char(20) not null

        create domain degree_level varchar(10)
                constraint degree_level_test
                        check (value in (’Bachelors’, ’Masters’, ’Doctorate’));

>> Create Table Extensions: Creating tables that have the same schema as an
existing table.

        create table temp_instructor like instructor

        create table t1 as
                (select *
                from instructor
                where dept_name= ’Music’)
        with data

>> Authorization
        >> Privilege
                Select(Read) : read access to relations or query in the views
                Insert
                Update
                Delete

        >> Forms of authorization to modify:
                Index - allows creation and deletion of indices.
                Resources - allows creation of new relations.
                Alteration - allows addition or deletion of attributes in a
relation.
                Drop - allows deletion of relations
        >> Grant: to confer authorization
                grant <privilege list>
                on <relation name or view name>
                to <user/role list>

                # <user list> is: 1. a user-id 2. public

                grant update on instructor to U1, U2, U3

                # The authorization may be to only some attributes but not on
specific tuples
                        grant update (name) on instructor to U1, U2, U3

        >> Revoke: to revoke authorization
                revoke <privilege list>
                on <relation name or view name>
                from <user/role list>

        >> Roles: Authorizations can be granted to roles as they are granted to
users
                create role lecturer;
                grant lecturer to U1;
                grant select on takes to lecturer;

                # U1 inherits all privileges of lecturer
                # We can have Chain of roles

        >> Views:
                # gives us the possibility to define authorization to some specific
tuples
                create view geo_instructor as
                        (select *
                        from instructor
                        where dept_name = ’Geology’);

                grant select on geo_instructor to  geo_staff

                Then a geo_staff member can issue: select * from geo_instructor;

        >> References: privilege to create foreign key
                grant reference (dept_name) on department to U1;

        >> Transfer
                >> with Grant option:
                        grant select on department to U1 with grant option;

                >> Cascade: is default
                        revoke select on department from U1, U2 cascade;
```

```
>> Restrict: prevent cascading revocation
        revoke select on department from U1, U2 restrict;
```