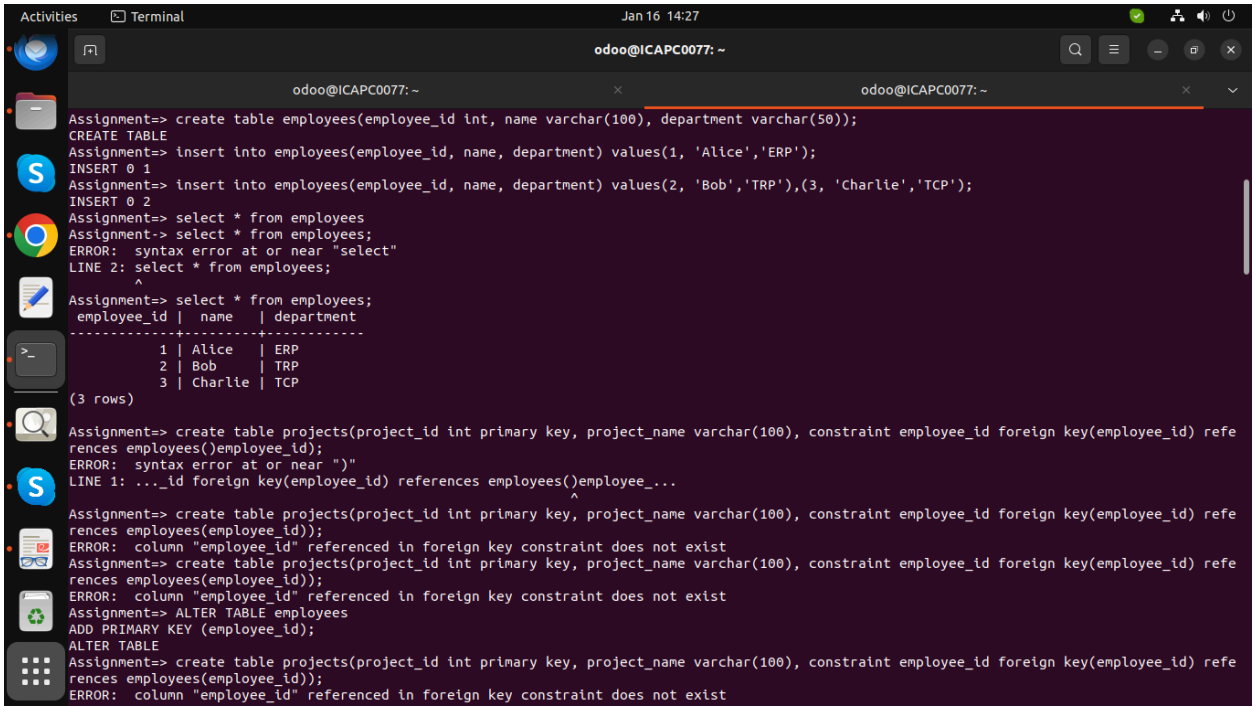
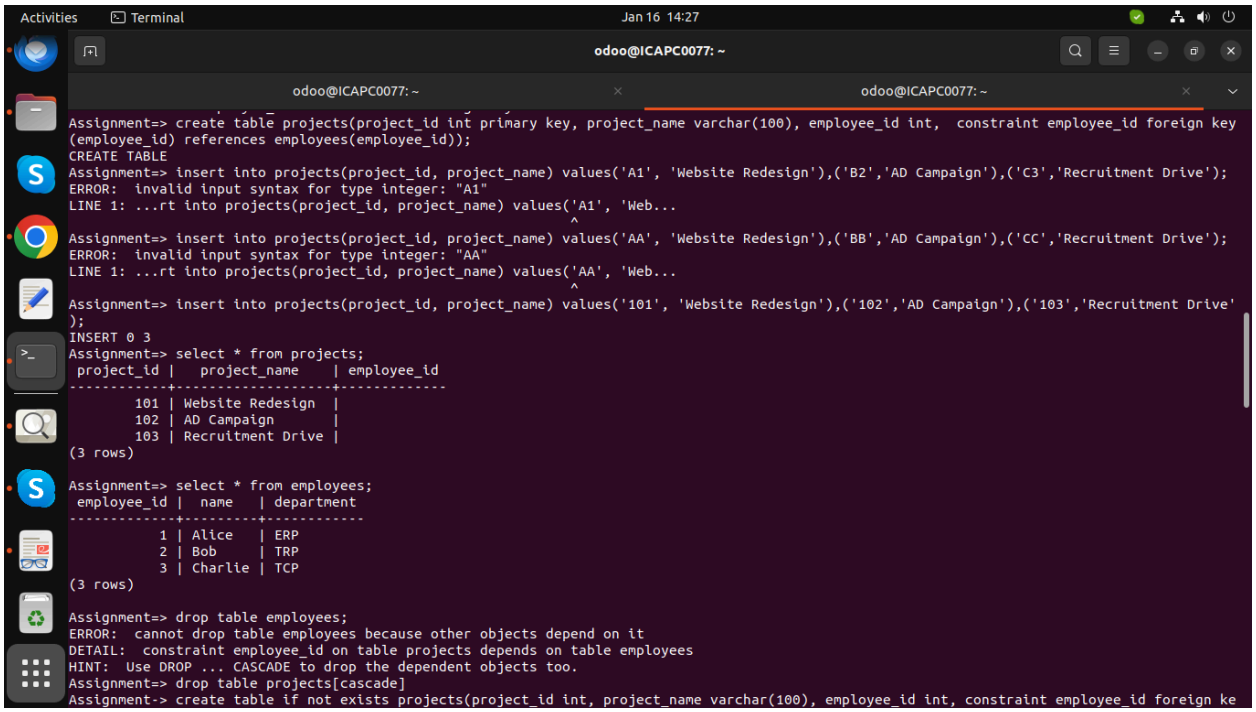
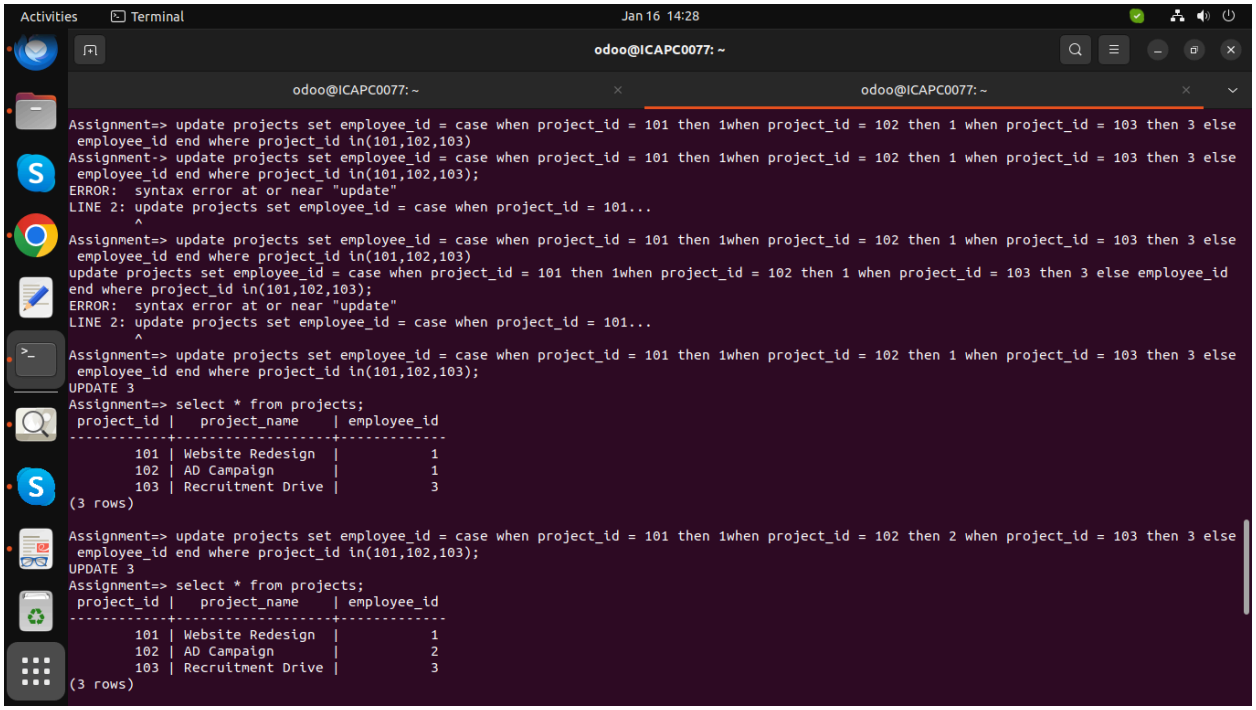


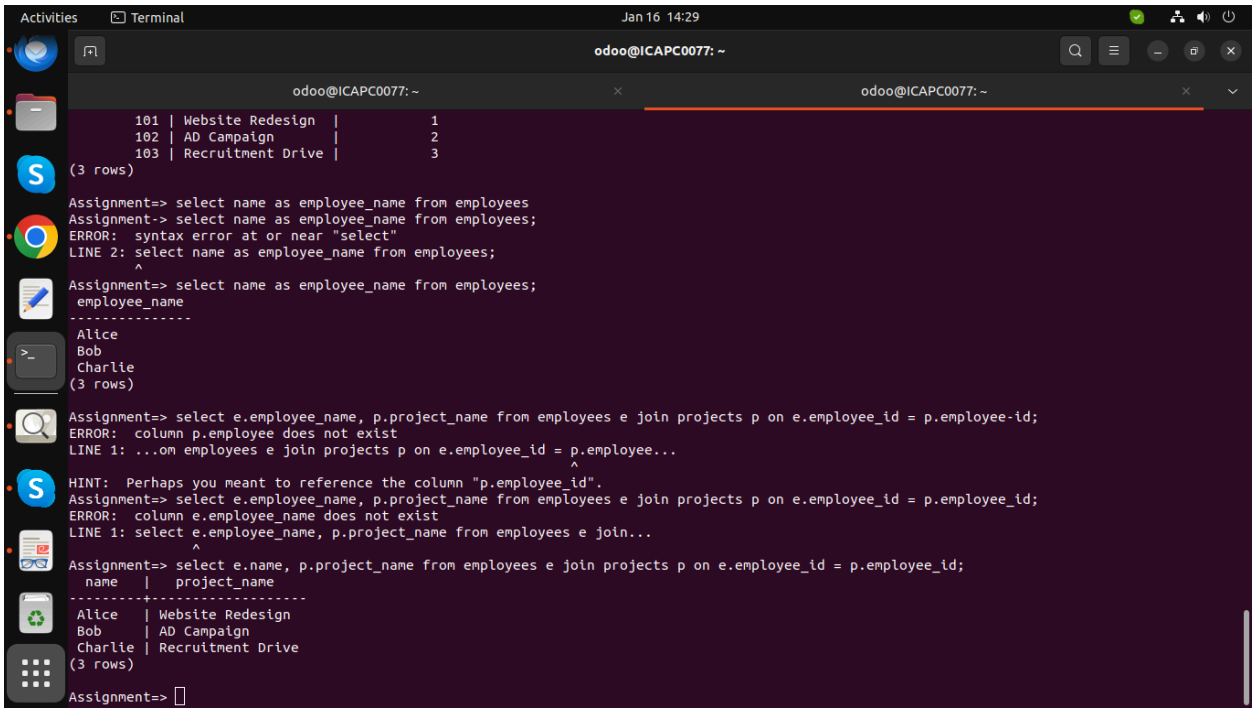

```
Assignment=> create table employees(employee_id int, name varchar(100), department varchar(50));
CREATE TABLE
Assignment=> insert into employees(employee_id, name, department) values(1, 'Alice','ERP');
INSERT 0 1
Assignment=> insert into employees(employee_id, name, department) values(2, 'Bob','TRP'),(3, 'Charlie','TCP');
INSERT 0 2
Assignment=> select * from employees
Assignment-> select * from employees;
ERROR:  syntax error at or near "select"
LINE 2: select * from employees;
        ^
Assignment=> select * from employees;
 employee_id |  name   | department
-------------+---------+------------
           1 | Alice   | ERP
           2 | Bob     | TRP
           3 | Charlie | TCP
(3 rows)

Assignment=> create table projects(project_id int primary key, project_name varchar(100), constraint employee_id foreign key(employee_id) refe
rences employees()employee_id);
ERROR:  syntax error at or near ")"
LINE 1: ..._id foreign key(employee_id) references employees()employee_...
                                                             ^
Assignment=> create table projects(project_id int primary key, project_name varchar(100), constraint employee_id foreign key(employee_id) refe
rences employees(employee_id));
ERROR:  column "employee_id" referenced in foreign key constraint does not exist
Assignment=> create table projects(project_id int primary key, project_name varchar(100), constraint employee_id foreign key(employee_id) refe
rences employees(employee_id));
ERROR:  column "employee_id" referenced in foreign key constraint does not exist
Assignment=> ALTER TABLE employees
ADD PRIMARY KEY (employee_id);
ALTER TABLE
Assignment=> create table projects(project_id int primary key, project_name varchar(100), constraint employee_id foreign key(employee_id) refe
rences employees(employee_id));
ERROR:  column "employee_id" referenced in foreign key constraint does not exist
```




```
Assignment=> create table projects(project_id int primary key, project_name varchar(100), employee_id int,  constraint employee_id foreign key
(employee_id) references employees(employee_id));
CREATE TABLE
Assignment=> insert into projects(project_id, project_name) values('A1', 'Website Redesign'),('B2','AD Campaign'),('C3','Recruitment Drive');
ERROR:  invalid input syntax for type integer: "A1"
LINE 1: ...rt into projects(project_id, project_name) values('A1', 'Web...
                                                             ^
Assignment=> insert into projects(project_id, project_name) values('AA', 'Website Redesign'),('BB','AD Campaign'),('CC','Recruitment Drive');
ERROR:  invalid input syntax for type integer: "AA"
LINE 1: ...rt into projects(project_id, project_name) values('AA', 'Web...
                                                             ^
Assignment=> insert into projects(project_id, project_name) values('101', 'Website Redesign'),('102','AD Campaign'),('103','Recruitment Drive'
);
INSERT 0 3
Assignment=> select * from projects;
 project_id |   project_name    | employee_id
------------+-------------------+-------------
        101 | Website Redesign  |
        102 | AD Campaign       |
        103 | Recruitment Drive |
(3 rows)

Assignment=> select * from employees;
 employee_id |  name   | department
-------------+---------+------------
           1 | Alice   | ERP
           2 | Bob     | TRP
           3 | Charlie | TCP
(3 rows)

Assignment=> drop table employees;
ERROR:  cannot drop table employees because other objects depend on it
DETAIL:  constraint employee_id on table projects depends on table employees
HINT:  Use DROP ... CASCADE to drop the dependent objects too.
Assignment=> drop table projects[cascade]
Assignment-> create table if not exists projects(project_id int, project_name varchar(100), employee_id int, constraint employee_id foreign ke
```




```
Assignment=> update projects set employee_id = case when project_id = 101 then 1when project_id = 102 then 1 when project_id = 103 then 3 else
 employee_id end where project_id in(101,102,103)
Assignment-> update projects set employee_id = case when project_id = 101 then 1when project_id = 102 then 1 when project_id = 103 then 3 else
 employee_id end where project_id in(101,102,103);
ERROR:  syntax error at or near "update"
LINE 2: update projects set employee_id = case when project_id = 101...
        ^
Assignment=> update projects set employee_id = case when project_id = 101 then 1when project_id = 102 then 1 when project_id = 103 then 3 else
 employee_id end where project_id in(101,102,103)
update projects set employee_id = case when project_id = 101 then 1when project_id = 102 then 1 when project_id = 103 then 3 else employee_id
end where project_id in(101,102,103);
ERROR:  syntax error at or near "update"
LINE 2: update projects set employee_id = case when project_id = 101...
        ^
Assignment=> update projects set employee_id = case when project_id = 101 then 1when project_id = 102 then 1 when project_id = 103 then 3 else
 employee_id end where project_id in(101,102,103);
UPDATE 3
Assignment=> select * from projects;
 project_id |    project_name     | employee_id
------------+---------------------+-------------
        101 | Website Redesign    |           1
        102 | AD Campaign         |           1
        103 | Recruitment Drive   |           3
(3 rows)

Assignment=> update projects set employee_id = case when project_id = 101 then 1when project_id = 102 then 2 when project_id = 103 then 3 else
 employee_id end where project_id in(101,102,103);
UPDATE 3
Assignment=> select * from projects;
 project_id |    project_name     | employee_id
------------+---------------------+-------------
        101 | Website Redesign    |           1
        102 | AD Campaign         |           2
        103 | Recruitment Drive   |           3
(3 rows)
```




```
        101 | Website Redesign    |           1
        102 | AD Campaign         |           2
        103 | Recruitment Drive   |           3
(3 rows)

Assignment=> select name as employee_name from employees
Assignment-> select name as employee_name from employees;
ERROR:  syntax error at or near "select"
LINE 2: select name as employee_name from employees;
        ^
Assignment=> select name as employee_name from employees;
 employee_name
---------------
 Alice
 Bob
 Charlie
(3 rows)

Assignment=> select e.employee_name, p.project_name from employees e join projects p on e.employee_id = p.employee-id;
ERROR:  column p.employee does not exist
LINE 1: ...om employees e join projects p on e.employee_id = p.employee...
                                                             ^
HINT:  Perhaps you meant to reference the column "p.employee_id".
Assignment=> select e.employee_name, p.project_name from employees e join projects p on e.employee_id = p.employee_id;
ERROR:  column e.employee_name does not exist
LINE 1: select e.employee_name, p.project_name from employees e join...
               ^
Assignment=> select e.name, p.project_name from employees e join projects p on e.employee_id = p.employee_id;
  name   |    project_name
---------+---------------------
 Alice   | Website Redesign
 Bob     | AD Campaign
 Charlie | Recruitment Drive
(3 rows)

Assignment=>
```

1. Difference between psql and sql?

Data Integrity and Concurrency:

PostgreSQL: Strong emphasis on data integrity with features like MVCC (Multi-Version Concurrency Control) and support for advanced data types like **JSON**

MySQL: It Prioritizes speed and performance which sometimes compromising strict **data integrity**. It has optional transactional support with **InnoDB**.

2. **Extensibility and Standards Compliance**:
    a. **PostgreSQL**: Highly extensible and compliant with SQL standards. It supports custom data types, operators, and full-text search.
    b. **MySQL**: Less extensible but simpler to use. It offers support for a variety of storage engines, which can be switched depending on the use case.

3. **Performance and Scalability**:
    a. **PostgreSQL**: Performs well with complex queries and large datasets, supporting horizontal scaling through sharding and replication.
    b. **MySQL**: Known for its read-heavy workload efficiency and high-speed read operations. MySQL's replication can be challenging in terms of **consistency**.
4. **Community and Support**:
    a. **PostgreSQL**: Has an active community, comprehensive documentation, and a wide variety of extensions.
    b. **MySQL**: Has a larger user base, with widespread commercial support and a variety of tools available.
5. **License**:
    a. **PostgreSQL**: Uses the PostgreSQL License, similar to the MIT License, which is permissive.
    b. **MySQL**: Uses the GNU General Public License (GPL) with commercial options available from Oracle.

Code of assignment

creatdb Assignment

```
create table employees(employee_id int, name varchar(100),
department varchar(50));

insert into employees(employee_id, name, department) values(1,
'Alice','ERP');

select * from employees;

ALTER TABLE employees
ADD PRIMARY KEY (employee_id);

create table projects(project_id int primary key, project_name
varchar(100), employee_id int,  constraint employee_id foreign
key(employee_id) references employees(employee_id));

insert into projects(project_id, project_name) values('101', 'Website
Redesign'),('102','AD Campaign'),('103','Recruitment Drive');

select * from projects;

update projects set employee_id = case when project_id = 101 then
1when project_id = 102 then 2 when project_id = 103 then 3 else
employee_id end where project_id in(101,102,103);

select e.name, p.project_name from employees e join projects p on
e.employee_id = p.employee_id;
```