Shell Trivia
My filesystem looks like the following:

```
$HOME/
  found/
    README.txt
  src/
    app.py
    templates/
      index.html
    app.old.py
  sql/
    create_tables.sql
  wsgi/
    found.wsgi
  logs/
    access_log
    error_log
```

1. What should I expect to see if I run the following commands:
   `cd $HOME/found`  
   `ls`  
   `README.txt`  
   `src`  
   `sql`

2. What should I expect to see if I run the following commands:
   `cd $HOME/found`  
   `cd src`  
   `cd ../sql`  
   `ls`  
   `create_tables.sql`

3. What should I expect to see if I run the following commands:
   `cd $HOME/found/src`  
   `rm app.old.py`  
   `ls`  
   `app.py`  
   `templates`
Git Trivia
1. I clone a repository with the following structure:

   $REPO/
   README.txt
dolores.py
maeve.py
bernard.py
ford.py

I add a new file, 'teddy.py', and make some changes to 'README.txt'. I then run the following command:

   git commit -am "Added Teddy to the project"
   git push

If someone else uses 'git clone', will their copy of the code include 'teddy.py'? Why or why not?

   No, I did not add teddy.py to the change set. -- only adds
   modified files, not new files.

2. I clone a repository with the following structure:

   $REPO/
   README.txt
airplane.py
car.py
truck.py

After cloning, I edit README.txt. Here are three potential sequences of commands I may run after editing README.txt:

   Case 1)
   git add README.txt
   git commit -m "README modifications"
   git push

   Case 2)
   git commit -am "README modifications"
   git push

   Case 3)
   git add README.txt
   git push
2.a. Do cases 1 and 2 produce the same result? If not, explain.

   Yes, both add README.txt to the change set, commit the change set, and push to the origin.

2.b. Do cases 2 and 3 produce the same result? If not, explain.

   No. Case 2 does not commit a change set at all and will probably generate a warning.

SQL Trivia
1a. Does the following diagram match the database created by the following create table statements? No.

   ![Diagram](image)

   ```sql
   CREATE TABLE cars (  
     car_pk    serial primary key,  
     color     varchar(17),  
     plate     varchar(10),  
     owner_fk  integer REFERENCES people(person_pk);  
   );

   CREATE TABLE owners (  
     car_fk    integer,  
     person_fk integer  
   );

   CREATE TABLE people (  
     person_pk serial primary key,  
     first     varchar(25),  
     last      varchar(25),  
     license   varchar(10)  
   );
   ```
1b. If the tables do not match the ERD, add the missing tables if any and cross out any unneeded lines in the existing create table statements. If the tables do match the ERD, draw a smiley face next to the ERD.

1c. Show what the records might look like for a red car jointly owned by Alex and Sam using the model after your edits, if any.

**Table Name: Cars**

<table>
<thead>
<tr>
<th>car_pk</th>
<th>color</th>
<th>plate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>XYZ 123</td>
</tr>
</tbody>
</table>

**Table Name: Owners**

<table>
<thead>
<tr>
<th>car_pk</th>
<th>Person_pk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table Name: People**

<table>
<thead>
<tr>
<th>Person_pk</th>
<th>first</th>
<th>last</th>
<th>license</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alex</td>
<td></td>
<td>TX 123</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Sam</td>
<td></td>
<td>TX 456</td>
<td></td>
</tr>
</tbody>
</table>

**Table Name:**
2. Given the following tables and data:

**Table Name: users**

<table>
<thead>
<tr>
<th>user_pk</th>
<th>username</th>
<th>role_fk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>skyler</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>adrian</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>taylor</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>river</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table Name: roles**

<table>
<thead>
<tr>
<th>role_pk</th>
<th>role</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>science officer</td>
</tr>
<tr>
<td>2</td>
<td>security officer</td>
</tr>
</tbody>
</table>

2a. What will the following queries return?

SELECT username FROM users WHERE user_pk=2;

- adrian

2b. SELECT username, role FROM users JOIN roles ON role_fk=role_pk WHERE user_pk=3;

- taylor, security officer
Risk Management
You have been tasked with implementing a web application with the following screen flow. Your dependences will all be installed but you are responsible for the database schema (script to create the tables), flask application code, and HTML templates.

- login allows the user to provide a username and password. The first time a username is seen by the application, a new user account will be automatically generated.
- results shows a listing of user accounts. The font and colors on the web page depend on the user profile.
- profile allows a user to change the font and color to be used on the results page.

1. There is some potential for technical risks in this project since you will be implementing a database schema, flask application code, HTML. The risks may be large (never worked with the technology before) or small (comfortable with the technology and have solved exactly this problem with the technology before). List these technologies in order from most risk to least risk. Explain why the risks are ordered this way relative to each other.

Highest risk - Flask Application code
Least experience with Flask

Middle risk - HTML
I haven't done much HTML recently and expect customer will want CSS as part of the solution

Lowest risk - SQL/database
I have a bit of experience with SQL and database needs seem simple
2. Break the project up into at least 3 incremental deliverables and no more than 5 incremental deliverables. For each incremental deliverable describe the demo that will be used to display progress (What will you show? How will you show it? What progress will it demonstrate?)

**Deliverable 1** - Login screen -> page with login name
Demonstrates that I have Flash working and can capture right username

**Deliverable 2** - Login -> Results screen
Demonstrates the user list is working

**Deliverable 3** - Login -> Profile
Demonstrates profile input can be saved/retrieved

**Deliverable 4** - Login -> Results w/prefs applied
Demonstrates complete functionality

**Deliverable 5** -

Why did you choose this order for incremental delivery?
The flow matches with the feature order.
I think the customer is most likely interested in the 1st 15 min.

3. What actions/steps the user should do to test that the application is working. Provide this as a list the pages the user should navigate to in order and what they should expect to see/do on each page. Specifically call out the steps that prove account login/creation work as well as the steps that prove the color/font changes work.

- Login screen - enter login and select profile screen
- Profile screen - set font and color and select result screen
- Result screen - verify that the colors selected appear on result screen
- Login screen - enter same login but wrong password, see rejection
- Login screen - enter same login with correct password and select results screen
Migration
Here is some HR data, showing people with their title and department, currently kept in a spreadsheet. One of the current problems the customer is having has to do with typos in titles and department names. Currently, each person is only allowed to have one title at a time. The customer renames departments often. Your team will be implementing a new web application to help reduce typos and make department renaming easier.

<table>
<thead>
<tr>
<th>first</th>
<th>last</th>
<th>username</th>
<th>role</th>
<th>department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oakley</td>
<td>Smith</td>
<td>soakley</td>
<td>vp</td>
<td>sales</td>
</tr>
<tr>
<td>Casey</td>
<td>Williams</td>
<td>cbill</td>
<td>director</td>
<td>sales</td>
</tr>
<tr>
<td>Hayden</td>
<td>Jones</td>
<td>hjones</td>
<td>vp</td>
<td>technology</td>
</tr>
</tbody>
</table>

1. The following three data models have been proposed to hold the HR data. Order the models (best, mid, worst) and for each model justify your rating. Your justification should mention design problems (e.g. model does not protect against typos or unneeded complexity) or benefits.

Model 1:

```
CREATE TABLE personnel (  
    first varchar(25),  
    last varchar(25),  
    username varchar(25),  
    role varchar(25),  
    department varchar(25)  
)
```

Rating and why: Worst, does not enforce a fixed number of role or department names. Doesn't really solve the typo problem.

Model 2: Best

```
CREATE TABLE person (  
    first varchar(25),  
    last varchar(25),  
    username varchar(25),  
    role varchar(25),  
    role_fk integer REFERENCES role(role_pk),  
    department_fk integer REFERENCES department(department_pk)  
)
```
CREATE TABLE role (  
    role_pk serial primary key,  
    role varchar(25)  
)  

CREATE TABLE department (  
    department_pk serial primary key,  
    department varchar(25)  
)

Rating and why: This model needed normalization.

Model 3: M.Id

CREATE TABLE person (  
    first varchar(25),  
    last varchar(25),  
    username varchar(25),  
    department_fk integer REFERENCES department(department_pk)  
)  

CREATE TABLE person_is (  
    username varchar(25) REFERENCES person(username),  
    role_fk integer REFERENCES role(role_pk)  
)  

CREATE TABLE role (  
    role_pk serial primary key,  
    role varchar(25)  
)  

CREATE TABLE department (  
    department_pk serial primary key,  
    department varchar(25)  
)

Rating and why: Mid - overly complex. Persons don't have multiple roles, so the join table is unnecessary.
2. The customer needs the spreadsheet data to be in the new system when the new system is brought online. The customer will correct all typos in the spreadsheet before cutover to the new system and would like automated scripts to transfer the data.

2a. For the model you have selected as the best, give the target table name and column name for each of the columns in the spreadsheet.

<table>
<thead>
<tr>
<th>Source Column</th>
<th>Table(column)</th>
</tr>
</thead>
<tbody>
<tr>
<td>first</td>
<td>person(first)</td>
</tr>
<tr>
<td>last</td>
<td>person(last)</td>
</tr>
<tr>
<td>username</td>
<td>person(username)</td>
</tr>
<tr>
<td>role</td>
<td>role(role)</td>
</tr>
<tr>
<td>department</td>
<td>department(department)</td>
</tr>
</tbody>
</table>

2b. Describe at a high level how your import script might work. How many passes through the spreadsheet and what is done on each pass (columns inserted)? If key values need to be looked up, when/how is that lookup done? Pseudo-code may be a good way to organize your solution.

In pass 1:
   for each row:
     r = row['role']
     if r exists in role table:
       skip
     else:
       insert r into role table

In pass 2:
   for each row:
     r = row['department']
     if r exists in department table:
       skip
     else:
       insert r into department table

In pass 3:
   for each row:
     look up role pk for row['role']
     look up department pk for row['department']
     insert the person record using the looked up role and department.