CS 325 Week 2-2

Continue database processing and development

More intro-to-SQL basics

Database processing and development

- DBMS Responsibilities: a DBMS, then, manages the database structure, controls access to the data stored in the db, and provides various tools to allow users/application programs to use the data

Database processing and development

- DBMS Responsibilities: a DBMS, then, manages the database structure, controls access to the data stored in the db, and provides various tools to allow users/application programs to use the data
- Database Application: Any program that uses a database, written in any language that can interact with a DBMS.

Database processing and development

- DBMS Responsibilities: a DBMS, then, manages the database structure, controls access to the data stored in the db, and provides various tools to allow users/application programs to use the data
- Database Application: Any program that uses a database, written in any language that can interact with a DBMS.
- Interacting with a DBMS: Examples include server-side JavaScript, Java servlets, and more.

Database processing and development

- DBMS Responsibilities: a DBMS, then, manages the database structure, controls access to the data stored in the db, and provides various tools to allow users/application programs to use the data
- Database Application: Any program that uses a database, written in any language that can interact with a DBMS.
- Interacting with a DBMS: Examples include server-side JavaScript, Java servlets, and more.
- DBMS Complexity: Simple DBMS, Multi-User DBMS, Large and Complex DBMS

Question:

What does "database design" focus on in the context of this course?

1) How the DBMS itself is built
2) How the DBMS manages the internal structure of files
3) Deciding which tables to create, their relationships, and what columns they should have
4) Designing the user interface for interacting with the database

Database processing and development

- What do we mean (in this course) by database design?
  - it is not how the DBMS is built, nor is it the file details about how the DBMS implements a database;

Database processing and development

- What do we mean (in this course) by database design?
  - it is not how the DBMS is built, nor is it the file details about how the DBMS implements a database;
  - we mean, instead, what tables we choose to create as part of a database? how are they interrelated? what are their columns?
    - this is what I mean, in the DB Reading Packet 1, by "the design of the database structure that will be used to store and manage data"

Database processing and development

- What do we mean (in this course) by database design?
  - With these powerful DBMSs, why do we care about database design?
  - Analogy: Having the most advanced OS, doesn’t make your a great programmer

Database processing and development

- What do we mean (in this course) by database design?
  - With these powerful DBMSs, why do we care about database design?
  - Analogy: Having the most advanced OS, doesn’t make your a great programmer
  - Even with the best DBMS, you can end up with a poorly designed database

Database processing and development

- What do we mean (in this course) by database design?
  - With these powerful DBMSs, why do we care about database design?
    - DBMS makes it easy to make tables? Why do I care what tables I pick, and how I relate them, and what columns I put in them, etc?
    - a wonderful DBMS does not mean that you can't build a database using it that will NOT work well for users and applications...
    - Even a good DBMS will perform poorly with a badly-designed db

A little bit of history about DBMS

- What is a File?
  - A file is commonly seen as a stream of characters or a collection of lines, which persists between programs.

A little bit of history about DBMS

- What is a File?
  - A file is commonly seen as a stream of characters or a collection of lines, which persists between programs.
- Historical Perspective

A little bit of history about DBMS

- What is a File?
  - A file is commonly seen as a stream of characters or a collection of lines, which persists between programs.
- Historical Perspective
- Mainframe Era

A little bit of history about DBMS

- What is a File?
  - A file is commonly seen as a stream of characters or a collection of lines, which persists between programs.
- Historical Perspective
- Mainframe Era - companies are using mainframes to manage their data through file processing systems

A little bit of history about DBMS

- What is a File?
  - A file is commonly seen as a stream of characters or a collection of lines, which persists between programs.
- Historical Perspective
- Mainframe Era - companies are using mainframes to manage their data through file processing systems
- important LIMITATIONS of file-processing systems as the quantity of data involved increases

A little bit of history about DBMS

- important LIMITATIONS of file-processing systems as the quantity of data involved increases:
  - separated and isolated data
    - It’s difficult to manage data when information is stored in separate files.

A little bit of history about DBMS

- important LIMITATIONS of file-processing systems as the quantity of data involved increases:
  - separated and isolated data
    - It’s difficult to manage data when information is stored in separate files.
    - Complex Queries and Inefficient Solutions

A little bit of history about DBMS

- important LIMITATIONS of file-processing systems as the quantity of data involved increases:
  - separated and isolated data
  - (unnecessary) data duplication
    - Data Duplication in Files

A little bit of history about DBMS

- important LIMITATIONS of file-processing systems as the quantity of data involved increases:
  - separated and isolated data
  - (unnecessary) data duplication
    - Data Duplication in Files
      - Think about university's student records

A little bit of history about DBMS

- important LIMITATIONS of file-processing systems as the quantity of data involved increases:
  - separated and isolated data
  - (unnecessary) data duplication
    - Data Duplication in Files
      - Think about university's student records
    - Efficiency and data integrity issues
      - Think about changing student phone number in one file

A little bit of history about DBMS

- important LIMITATIONS of file-processing systems as the quantity of data involved increases:
  - separated and isolated data
  - (unnecessary) data duplication
  - application program dependency

A little bit of history about DBMS

- important LIMITATIONS of file-processing systems as the quantity of data involved increases:
  - separated and isolated data
  - (unnecessary) data duplication
  - application program dependency
  - difficulty of representing data in the users' perspective

A little bit of history about DBMS

- important LIMITATIONS of file-processing systems as the quantity of data involved increases:
  - separated and isolated data
  - (unnecessary) data duplication
  - application program dependency
  - difficulty of representing data in the users' perspective
    - Think about that user often just want to see a subset of the data

Question

Which of the following can happen because of separated and isolated data as the number of files (and the amount of data stored in those files) increases?

1) can make it easier to keep data up to date and consistent
2) can have a tendency to increase application program independence
3) can make integrating the data more practical
4) can make data in different files difficult to relate to each other when needed/desired


HAMBURGERS
Arcata
local • seasonal • organic
folie douce


Welcome
$1
$1.50


ARCATA
BEER
PIZZA


SLICE OF HUMBOLDT PIE


Carl's Jr.
NOW HIRING


DUTCHY'S PIZZA
1116
OPEN


FOR LEASE

Restaurants Example

cheap-restaurants.txt — Edited

```
Stars Hamburgers - Arcata
Taco Bell - McKinleyville
McDonald's - Valley West
Arcata Pizza Deli - Arcata - if get slice-of-pizza
special
Philly Cheese Steak Shoppe - Arcata
Hey Juan Burritos - Arcata
Slice of Humboldt Pie - Arcata
Carl's Jr - Valley West
Dutchy's Pizza - Arcata
Wendy's - Eureka
McDonald's - Eureka
Burger King - McKinleyville
Subway - Arcata
```

have-burgers

View Zoom Add Page Insert Table Chart Text Shape Media Comment

Burger King: Arcata, 839-9299
Arcata Pizza & Deli: Arcata, 822-4650
McDonald's: Arcata, 822-0888
Stars Burgers: Eureka, 445-2061
Wendy's: Eureka, 441-4900
The Burger Joint: Arcata, 630-5144
Humboldt Brews: Arcata, 826-2739
Toni's 24 Hr Restaurant: Arcata, 822-0091

cheap-restaurants.txt

have-burgers.docx

pizza-available.ods

sandwich-shops.odt

pizza-available.ods

Arial 10 A5 Marcelli's

	A	B	C
1	Mazzotti's	Arcata	773 8th Street
2	Dutchy's	Arcata	1116 11th Street
3	La Trattoria	Arcata	30 Sunnybrae Ctr
4	Arcata Pizza Deli	Arcata	1057 H St
5	Marcelli's	Eureka	1323 5th Street
6	Paul's Live from New York	Arcata	665 Samoa Blvd

sandwich-shops.odt

12

McDonald's, 822-0888, open 24 hours
Wendy's, 707-441-4900, 10:00 am - 1:00 am every day
Philly Cheese Steak, 825-7400, 11 am - 9 pm daily
Stars Hamburgers, (707) 826-1379, M-Sat 11:00 am to 8:00 pm, Sun noon to 6:00 pm
Subway, 840-9811, don't know
Arcata Pizza Deli, 707 822-4650, 11:00 am - at least 12:00 am (later on some days)
Toni's, 822-0091, 24/7

Restaurants Example

- Categorizing Data in Files
- Challenges with Separated and Isolated Data

Restaurants Example

- Categorizing Data in Files
- Challenges with Separated and Isolated Data
  - Pulling data from multiple files is difficult and time-consuming.

Restaurants Example

- Categorizing Data in Files
- Challenges with Separated and Isolated Data
- Data Duplication Issues

Restaurants Example

- Categorizing Data in Files
- Challenges with Separated and Isolated Data
- Data Duplication Issues
- The Bigger Picture
  - Files aren’t inherently bad, but as data grows, managing it becomes harder.
  - Manually checking data isn’t feasible with hundreds of rows.
  - This is why people seek alternatives to files.

A little bit of history about DBMS

- important LIMITATIONS of file-processing systems as the quantity of data involved increases:
  - separated and isolated data
  - (unnecessary) data duplication
  - application program dependency
  - difficulty of representing data in the users' perspective
- these helped fuel a desire for something better!...database processing systems;
- seeking to IMPROVE on the situation, at least potentially

A little bit of history about DBMS

- seeking to IMPROVE on the situation, at least potentially
  - potential for more-integrated data

A little bit of history about DBMS

- seeking to IMPROVE on the situation, at least potentially
  - potential for more-integrated data
  - potential for less unnecessary data duplication

A little bit of history about DBMS

- seeking to IMPROVE on the situation, at least potentially
  - potential for more-integrated data
  - potential for less unnecessary data duplication
  - potential for decreased application program dependency

A little bit of history about DBMS

- seeking to IMPROVE on the situation, at least potentially
  - potential for more-integrated data
  - potential for less unnecessary data duplication
  - potential for decreased application program dependency
    - Think about Object-Oriented Programming or when you call a sorting method

A little bit of history about DBMS

- seeking to IMPROVE on the situation, at least potentially
  - potential for more-integrated data
  - potential for less unnecessary data duplication
  - potential for decreased application program dependency
    - Think about Object-Oriented Programming or when you call a sorting method
    - SQL allows you to write queries that often remain unchanged even if the DBMS changes.

A little bit of history about DBMS

- seeking to IMPROVE on the situation, at least potentially
  - potential for more-integrated data
  - potential for less unnecessary data duplication
  - potential for decreased application program dependency
  - potential for easier representation of the user’s perspectives

Question:

How does a well-designed database help minimize data duplication?

1) It stores every detail in every table to ensure completeness.
2) It keeps duplication to a minimum by only repeating data necessary to link tables together.
3) It completely eliminates the need to store any duplicate data.
4) It duplicates all data to ensure faster access.

A little bit of history about DBMS

- seeking to IMPROVE on the situation, at least potentially
  - potential for more-integrated data
  - potential for less unnecessary data duplication
  - potential for decreased application program dependency
  - potential for easier representation of the user's perspectives
- database system
  - Centralized Data Storage
  - No Need for Manual Consolidation

A little bit of history about DBMS

- seeking to IMPROVE on the situation, at least potentially
  - potential for more-integrated data
  - potential for less unnecessary data duplication
  - potential for decreased application program dependency
  - potential for easier representation of the user's perspectives
- database system
  - Centralized Data Storage
  - No Need for Manual Consolidation
  - Minimal Data Duplication

```sql
-- quickie table(s) for in-class restaurant example

-- (warning: for this example, I didn't really MODEL this
--    scenario properly!!)

drop table restaurant cascade constraints;

create table restaurant
(rest_id         integer,
 rest_name       varchar2(50),
 rest_str_addr   varchar2(50),
 rest_city       varchar2(30),
 rest_phone      varchar2(12),
 has_pizza       char(1) check(has_pizza in ('y', 'n')),
 has_burgers     char(1) check(has_burgers in ('y', 'n')),
 has_sandwiches  char(1) check(has_sandwiches in ('y', 'n')),
 primary key     (rest_id)
);

drop table rest_menu_faves cascade constraints;

create table rest_menu_faves
(rest_id         integer,
 menu_item       varchar2(100),
 menu_item_cost  decimal(5, 2),
 primary key (rest_id, menu_item),
 foreign key (rest_id) references restaurant
);
```

```sql
insert into restaurant
values
(1, 'Stars Hamburgers', '1535 G St', 'Arcata', '707-826-1379', 'n', 'y', 'y');

insert into restaurant
values
(2, 'Taco Bell', '1811 Central Ave', 'Mckinleyville', '707-839-7734',
    'n', 'n', 'n');

insert into restaurant
values
(3, 'McDonald''s', '4901 Valley West Blvd', 'Arcata', '707-822-0888',
    'n', 'y', 'y');

insert into restaurant
values
(4, 'Arcata Pizza Deli', '1057 H St', 'Arcata', '707-822-4650', 'y', 'y', 'y');

insert into restaurant
values
(5, 'Philly Cheese Steak Shoppe', '1811 G St', 'Arcata', '707-825-7400',
    'n', 'n', 'y');

insert into restaurant
values
(6, 'Hey Juan Burritos', '1642 1/2 G St', 'Arcata', '707-822-8433', 'n',
    'n', 'n');

insert into restaurant
values
(7, 'Slice of Humboldt Pie', '828 I St', 'Arcata', '707-630-5100', 'n',
     'n', 'n');
```

```
column rest_name format a20 tru
column rest_str_addr format a15 tru

-- which restaurants serve burgers AND pizza?

prompt Restaurants that serve burgers AND pizza:
prompt ==========

select rest_name
from   restaurant
where  has_pizza = 'y'
       and has_burgers = 'y';

-- phone numbers of restaurants that serve pizza?

prompt phone numbers of restaurants that serve pizza:
prompt ==========

select rest_name, rest_phone
from   restaurant
where  has_pizza = 'y';
```

```
-- addresses of burger places?

prompt addresses of burger places:
prompt ==========

select rest_name, rest_str_addr, rest_city
from   restaurant
where  has_burgers = 'y';

-- restaurants in Eureka that serve sandwiches?

prompt restaurants in Eureka that serve sandwiches
prompt ==========

select rest_name, rest_city
from   restaurant
where  rest_city = 'Eureka'
       and has_sandwiches = 'y';
```

A little bit of history about DBMS

- obviously, a database-processing approach CAN have disadvantages, also;
  - DBMSs are complex and can require a lot of memory.

A little bit of history about DBMS

- obviously, a database-processing approach CAN have disadvantages, also;
  - DBMSs are complex and can require a lot of memory.
  - Costs can be high, both for the DBMS itself and for converting data.
  - Performance may not always be better than file-based systems, especially for highly optimized custom applications.

A little bit of history about DBMS

- obviously, a database-processing approach CAN have disadvantages, also;
  - DBMSs are complex and can require a lot of memory.
  - Costs can be high, both for the DBMS itself and for converting data.
  - Performance may not always be better than file-based systems, especially for highly optimized custom applications.
  - When streaming large amounts of data, DBMS overhead can slow down processing.
  - Centralized data introduces the risk of a single point of failure, requiring careful attention to backups and security.

Question

Which of the following should have the least unnecessary data duplication?

1) a spreadsheet system
2) a file-processing system
3) a database processing system
4) a rolodex system

MORE intro to SQL and SQL*Plus

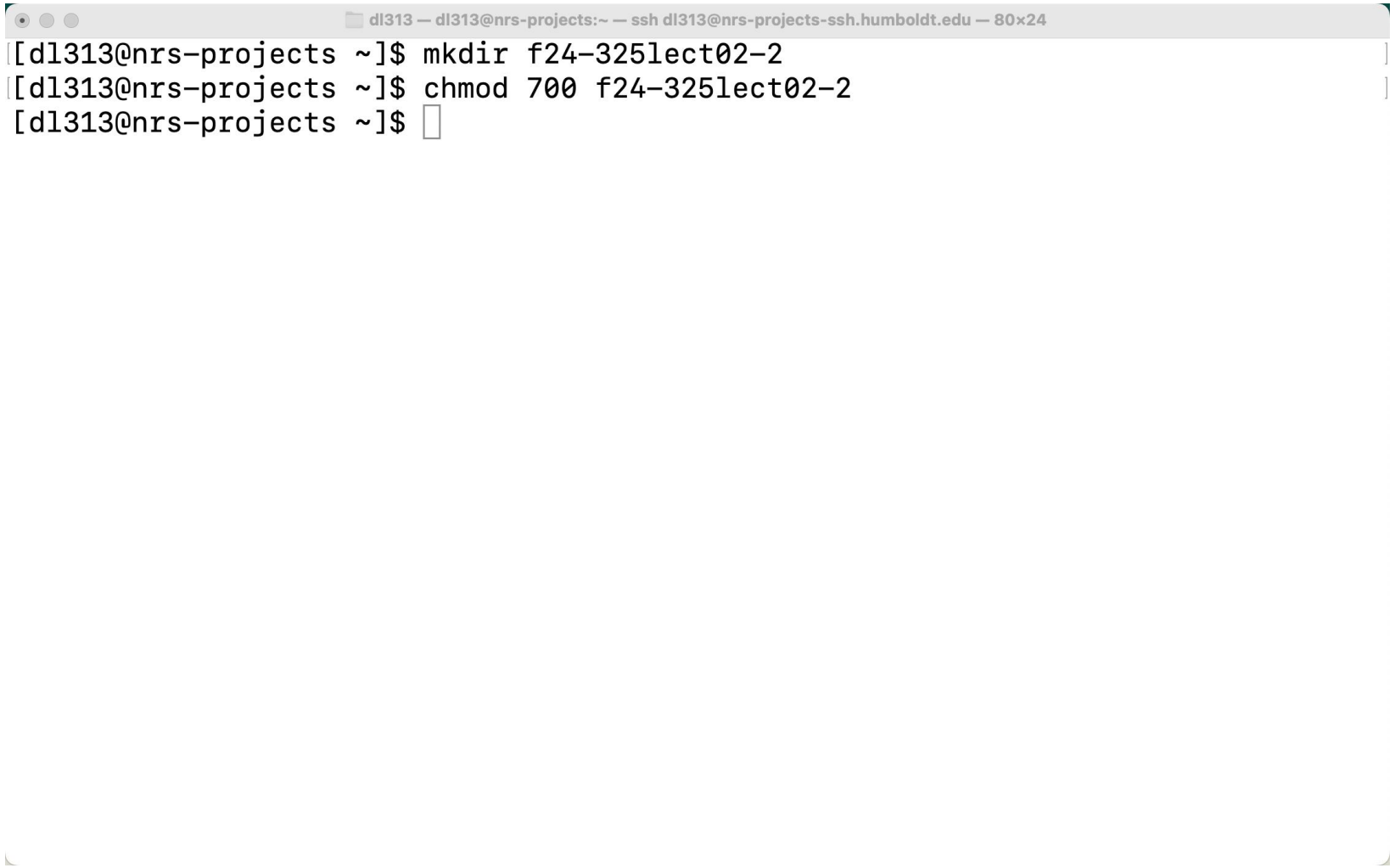
- SQL*Plus command: prompt
  - just prints a message to the screen (or, say, to spooled output)
  - JUST follow the command with the desired message (!)
  - if you follow it with nothing? You get a blank line to the screen...!

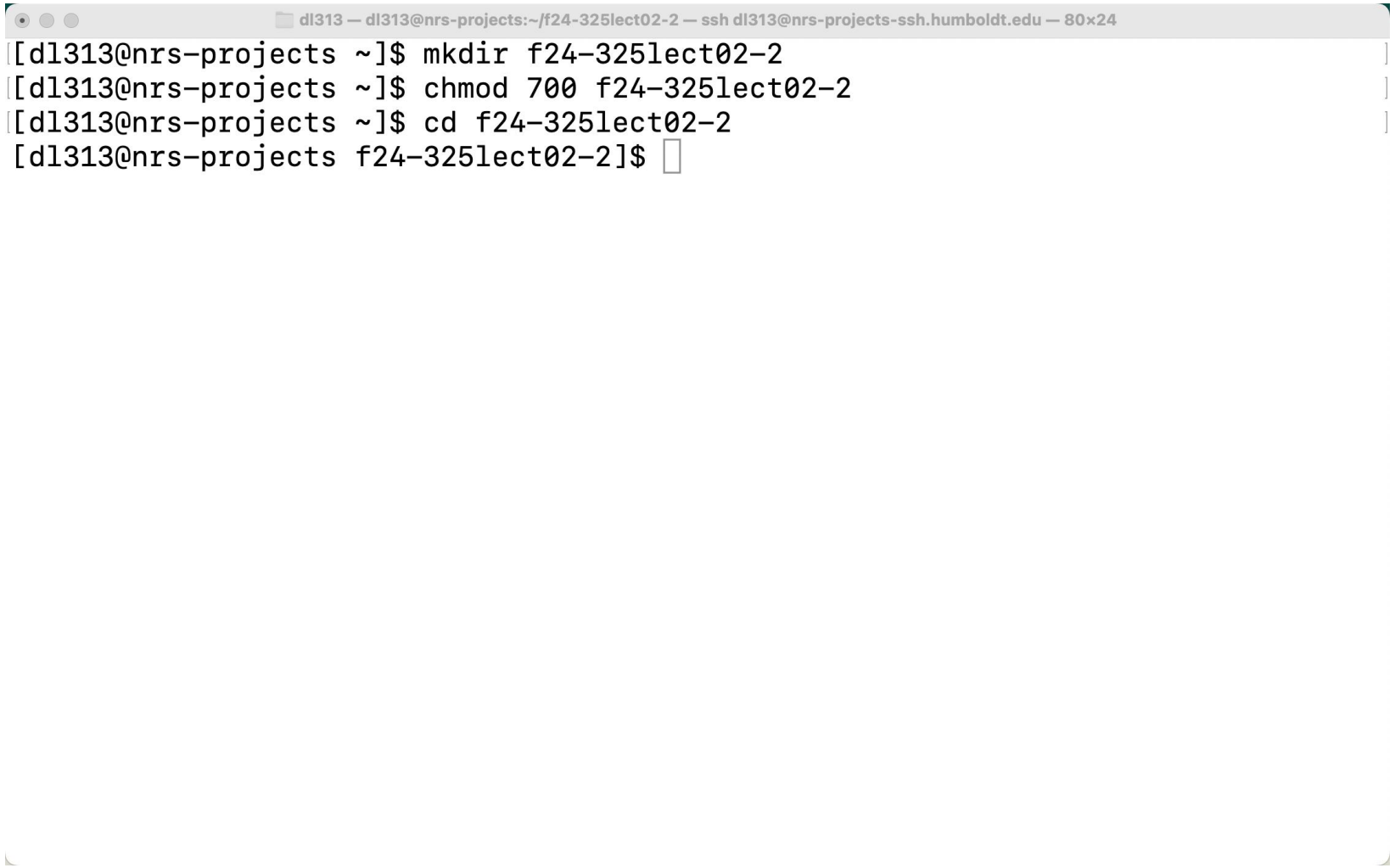
MORE intro to SQL and SQL*Plus

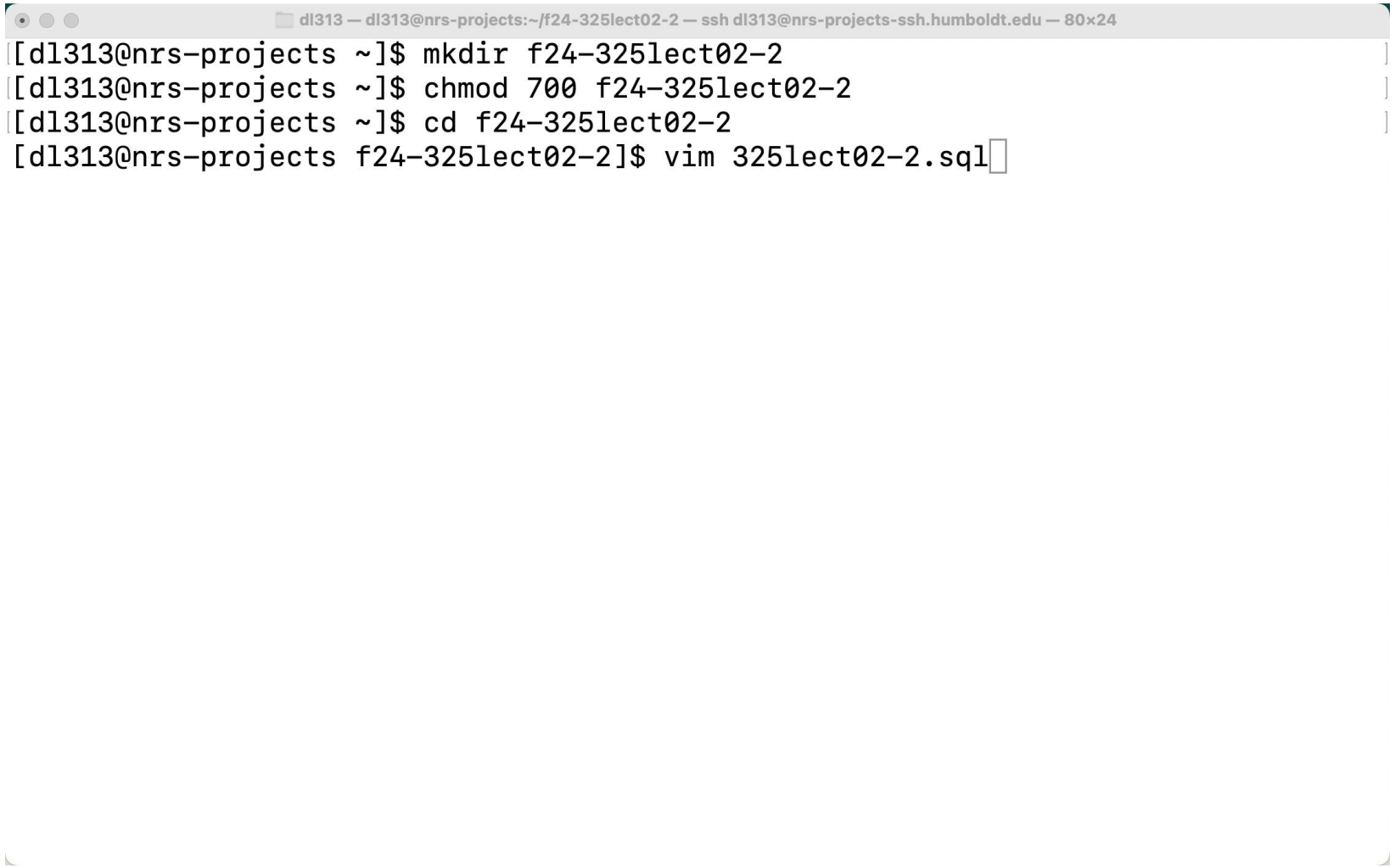
- SQL*Plus command: prompt
  - just prints a message to the screen (or, say, to spooled output)
  - JUST follow the command with the desired message (!)
  - if you follow it with nothing? You get a blank line to the screen...!

```
prompt
prompt Howdy there CS 325
prompt
```


dl313 — dl313@nrs-projects:~ — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
[dl313@nrs-projects ~]$ mkdir f24-325lect02-2
[dl313@nrs-projects ~]$


dl313 — dl313@nrs-projects:~ — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
[dl313@nrs-projects ~]$ mkdir f24-325lect02-2
[dl313@nrs-projects ~]$ chmod 700 f24-325lect02-2
[dl313@nrs-projects ~]$

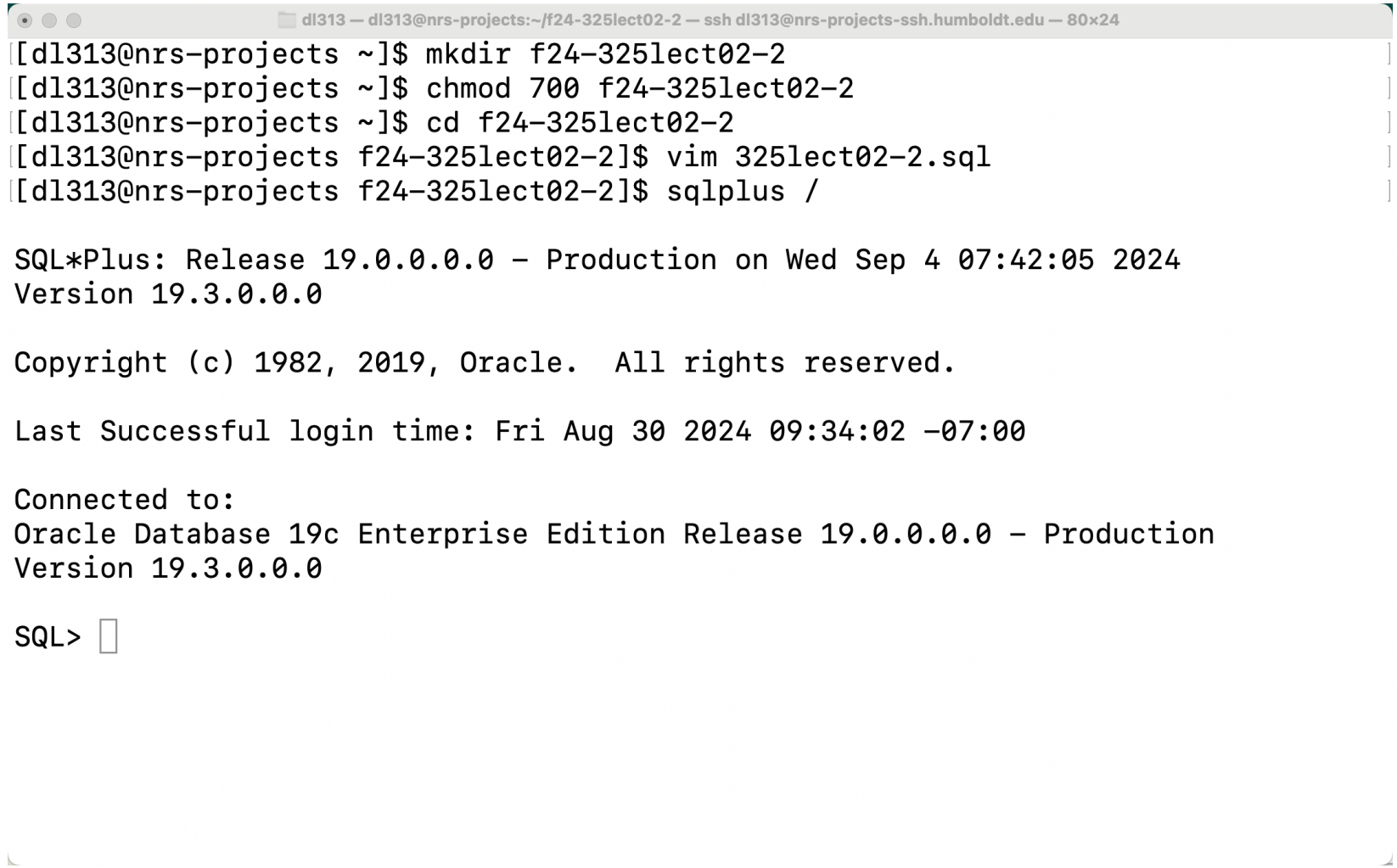

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
[dl313@nrs-projects ~]$ mkdir f24-325lect02-2
[dl313@nrs-projects ~]$ chmod 700 f24-325lect02-2
[dl313@nrs-projects ~]$ cd f24-325lect02-2
[dl313@nrs-projects f24-325lect02-2]$

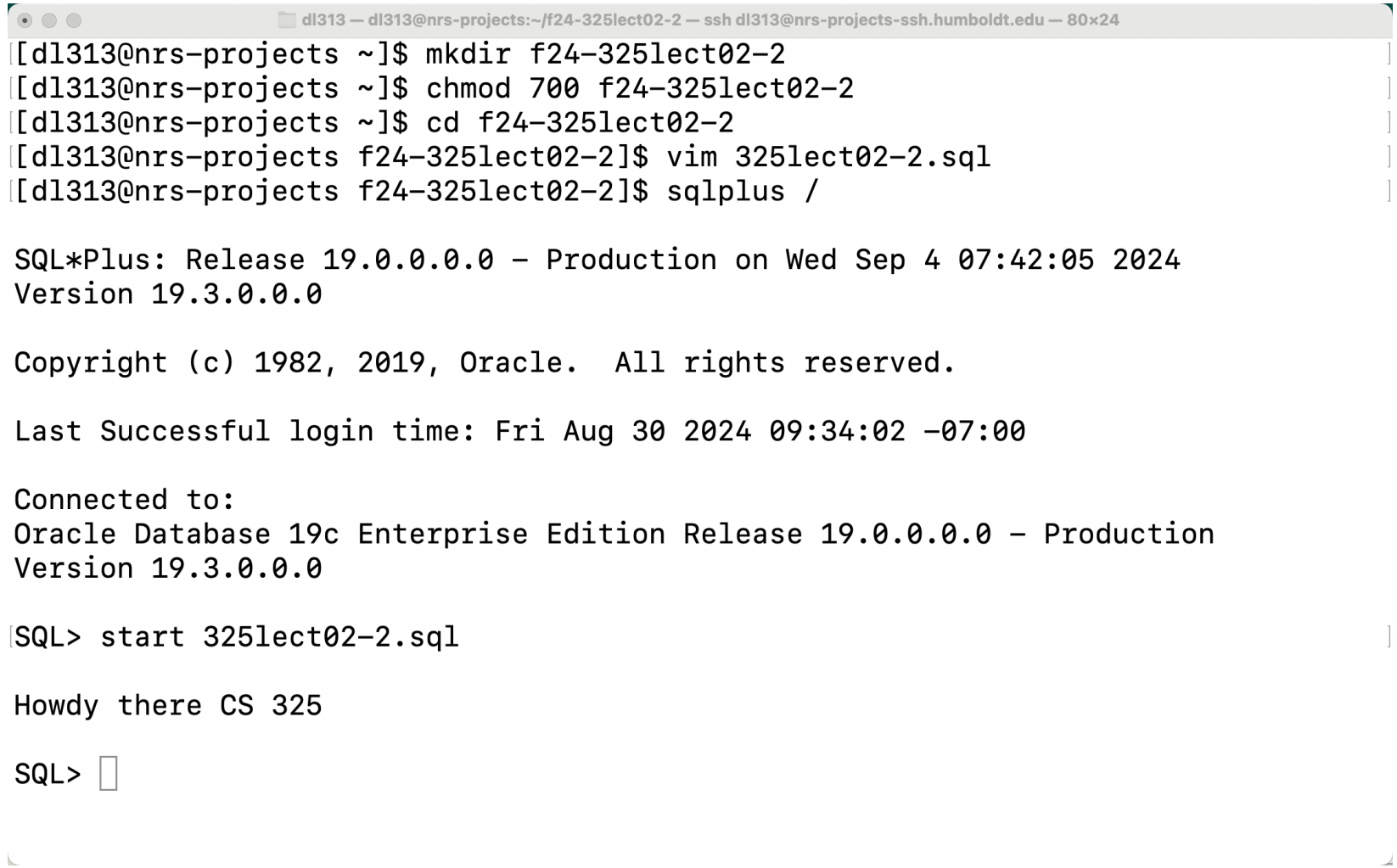

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
[dl313@nrs-projects ~]$ mkdir f24-325lect02-2
[dl313@nrs-projects ~]$ chmod 700 f24-325lect02-2
[dl313@nrs-projects ~]$ cd f24-325lect02-2
[dl313@nrs-projects f24-325lect02-2]$ vim 325lect02-2.sql

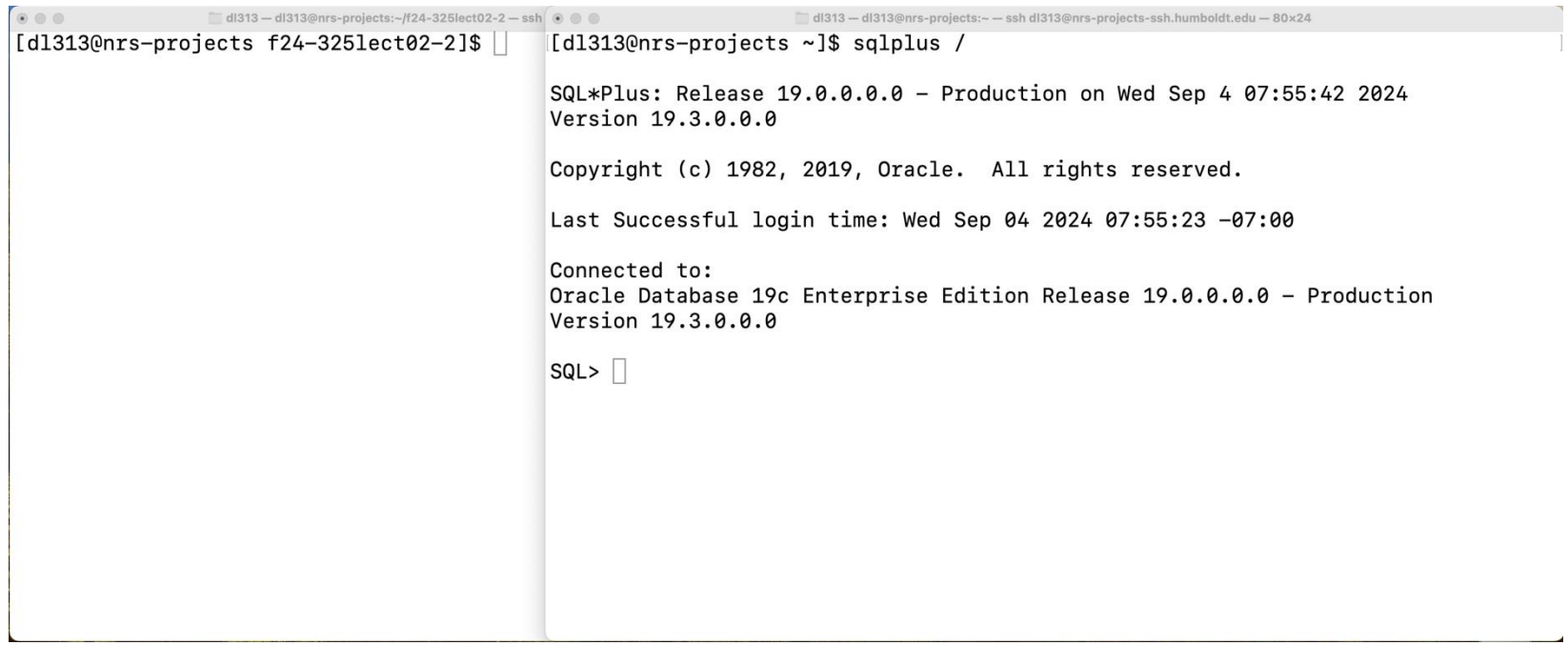

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
-- demo of the prompt
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --
1,22
All


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
-- demo of the prompt
prompt
prompt Howdy there CS 325
prompt
-- INSERT --
5,7
All


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
-- demo of the prompt
prompt
prompt Howdy there CS 325
prompt
:wq


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
[dl313@nrs-projects ~]$ mkdir f24-325lect02-2
[dl313@nrs-projects ~]$ chmod 700 f24-325lect02-2
[dl313@nrs-projects ~]$ cd f24-325lect02-2
[dl313@nrs-projects f24-325lect02-2]$ vim 325lect02-2.sql
[dl313@nrs-projects f24-325lect02-2]$ sqlplus /
SQL*Plus: Release 19.0.0.0.0 - Production on Wed Sep 4 07:42:05 2024
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle. All rights reserved.
Last Successful login time: Fri Aug 30 2024 09:34:02 -07:00
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
SQL>


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
[dl313@nrs-projects ~]$ mkdir f24-325lect02-2
[dl313@nrs-projects ~]$ chmod 700 f24-325lect02-2
[dl313@nrs-projects ~]$ cd f24-325lect02-2
[dl313@nrs-projects f24-325lect02-2]$ vim 325lect02-2.sql
[dl313@nrs-projects f24-325lect02-2]$ sqlplus /
SQL*Plus: Release 19.0.0.0.0 - Production on Wed Sep 4 07:42:05 2024
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle. All rights reserved.
Last Successful login time: Fri Aug 30 2024 09:34:02 -07:00
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
SQL> start 325lect02-2.sql
Howdy there CS 325
SQL>


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh
[dl313@nrs-projects f24-325lect02-2]$
dl313 — dl313@nrs-projects:~ — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
[dl313@nrs-projects ~]$ sqlplus /
SQL*Plus: Release 19.0.0.0.0 - Production on Wed Sep 4 07:55:42 2024
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle. All rights reserved.
Last Successful login time: Wed Sep 04 2024 07:55:23 -07:00
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
SQL>

MORE intro to SQL and SQL*Plus

- SQL*Plus command: prompt
  - just prints a message to the screen (or, say, to spooled output)
  - JUST follow the command with the desired message (!)
  - if you follow it with nothing? You get a blank line to the screen...!
- SQL*Plus command: / command

MORE intro to SQL and SQL*Plus

- SQL*Plus command: prompt
  - just prints a message to the screen (or, say, to spooled output)
  - JUST follow the command with the desired message (!)
  - if you follow it with nothing? You get a blank line to the screen...!
- SQL*Plus command: / command
  - if you have, in SQL*Plus, a command that is just: “/”

    ... this redoes the previous SQL statement


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
[dl313@nrs-projects f24-325lect02-2]$ vim 325lect02-2.sql

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
-- demo of the prompt

prompt
prompt Howdy there CS 325
prompt

-- reminder: this version of a select statement displays
-- all the rows and all the columns of the named table

select *
from student;

~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                    12,1          All
```

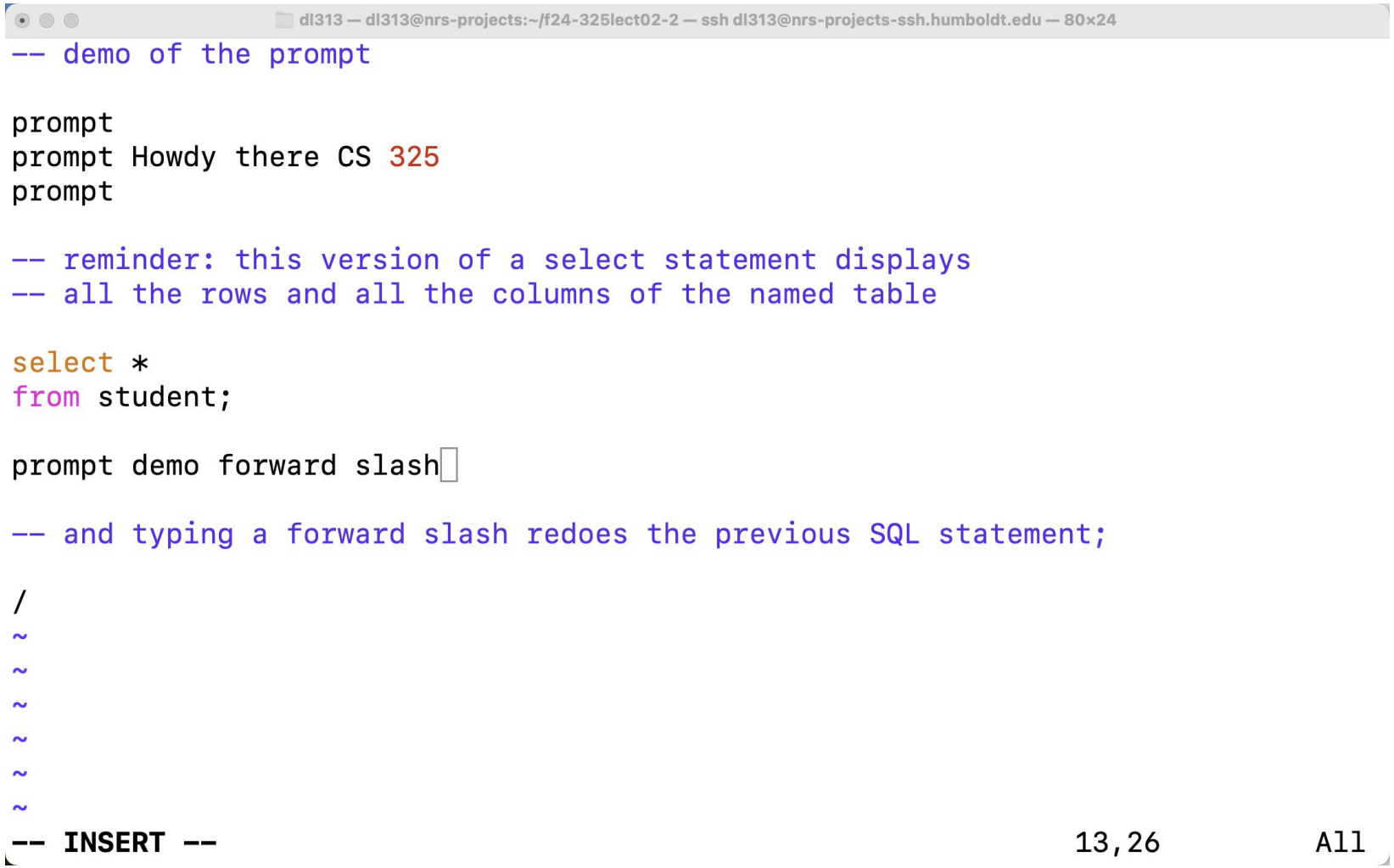
```
dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
[dl313@nrs-projects ~]$ cd f24-325lect02-2/
[dl313@nrs-projects f24-325lect02-2]$ sqlplus /

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Sep 4 08:18:40 2024
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Last Successful login time: Wed Sep 04 2024 07:55:42 -07:00

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> start 325lect02-2.sql

Howdy there CS 325


STUDENT_ID STUDENT_NAME         STUDENT_SHIRTCOLOR
---------- -------------------- --------------------
       313 Dongcheng Li         black

SQL>
```

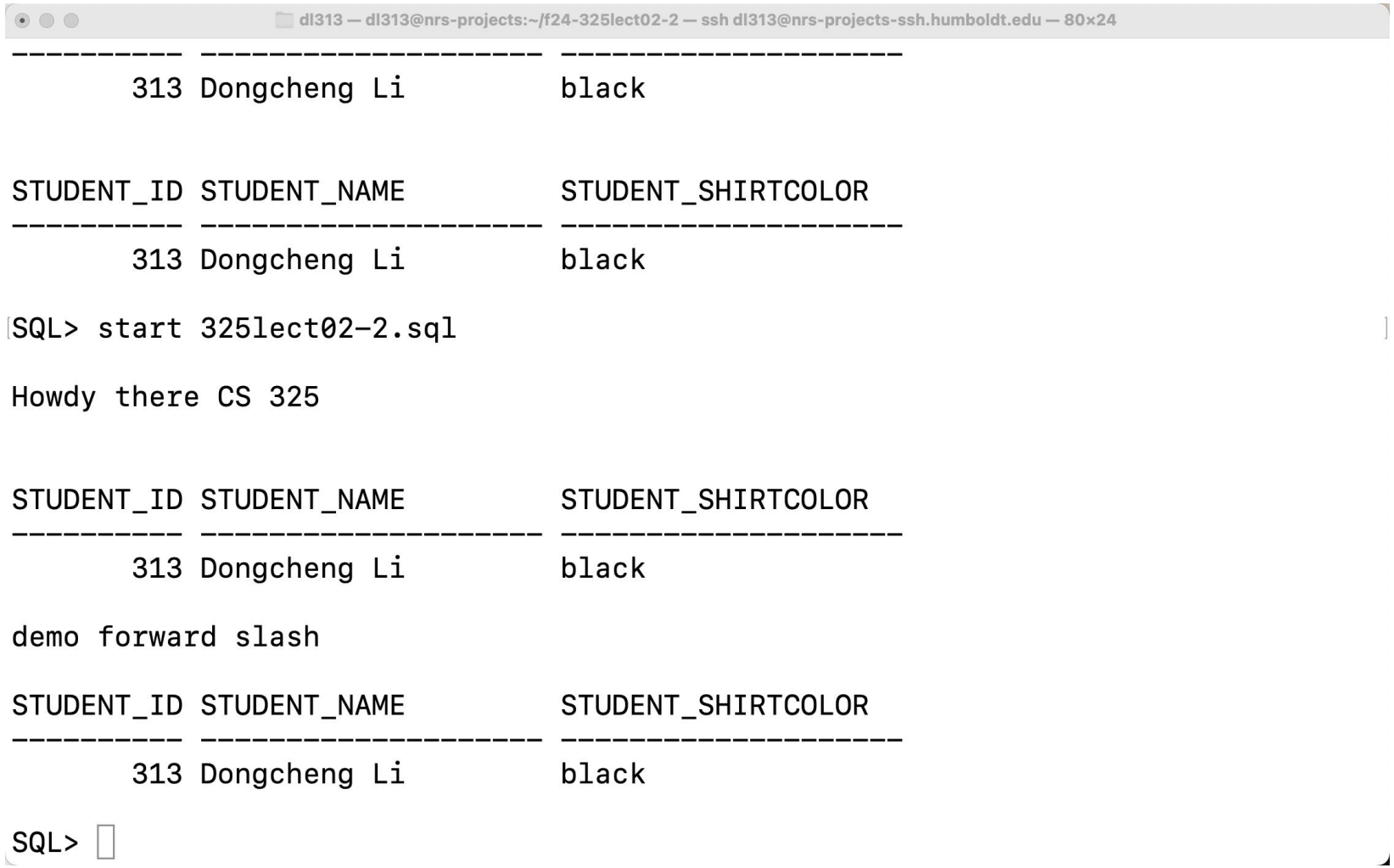
dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
-- demo of the prompt

prompt
prompt Howdy there CS 325
prompt

-- reminder: this version of a select statement displays
-- all the rows and all the columns of the named table

select *
from student;

-- and typing a forward slash redoes the previous SQL statement;

/
~
~
~
~
~
~
~
~
-- INSERT --                                    15,2          All
```

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Last Successful login time: Wed Sep 04 2024 08:18:40 -07:00

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> start 325lect02-2.sql

Howdy there CS 325


STUDENT_ID STUDENT_NAME         STUDENT_SHIRTCOLOR
---------- -------------------- --------------------
       313 Dongcheng Li         black


STUDENT_ID STUDENT_NAME         STUDENT_SHIRTCOLOR
---------- -------------------- --------------------
       313 Dongcheng Li         black

SQL>
```


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
-- demo of the prompt
prompt
prompt Howdy there CS 325
prompt
-- reminder: this version of a select statement displays
-- all the rows and all the columns of the named table
select *
from student;
prompt demo forward slash
-- and typing a forward slash redoes the previous SQL statement;
/
~
~
~
~
~
~
-- INSERT --
13,26
All


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
---------- -------------------- --------------------
313 Dongcheng Li black
STUDENT_ID STUDENT_NAME STUDENT_SHIRTCOLOR
---------- -------------------- --------------------
313 Dongcheng Li black
SQL> start 325lect02-2.sql
Howdy there CS 325
STUDENT_ID STUDENT_NAME STUDENT_SHIRTCOLOR
---------- -------------------- --------------------
313 Dongcheng Li black
demo forward slash
STUDENT_ID STUDENT_NAME STUDENT_SHIRTCOLOR
---------- -------------------- --------------------
313 Dongcheng Li black
SQL>

MORE intro to SQL and SQL*Plus

- SQL*Plus command: prompt
  - just prints a message to the screen (or, say, to spooled output)
  - JUST follow the command with the desired message (!)
  - if you follow it with nothing? You get a blank line to the screen...!
- SQL*Plus command: / command
  - if you have, in SQL*Plus, a command that is just: “/”

    ... this redoes the previous SQL statement
- more discussion of a FEW more column types (we'll call these column DOMAINS or attribute DOMAINS next week...) available in SQL

MORE intro to SQL and SQL*Plus

- more discussion of a FEW more column types (we'll call these column DOMAINS or attribute DOMAINS next week...) available in SQL
- basic SQL create table statement syntax:

```
create table new_table_name
(col_name col_type any_additional_constraints,
 col_name col_type any_additional_constraints,
 ...
 primary key (col_name, col_name, ...),
 foreign key (col_name) references another_table
);
```

MORE intro to SQL and SQL*Plus

- more discussion of a FEW more column types (we'll call these column DOMAINS or attribute DOMAINS next week...) available in SQL
- basic SQL create table statement syntax.
- here are a FEW of the more-common data types that the Oracle DBMS supports for columns in its implementation of SQL:

MORE intro to SQL and SQL*Plus

- here are a FEW of the more-common data types that the Oracle DBMS supports for columns in its implementation of SQL:
  - varchar2(n) - a varying-length character string up to n characters long

MORE intro to SQL and SQL*Plus

- here are a FEW of the more-common data types that the Oracle DBMS supports for columns in its implementation of SQL:
  - varchar2(n) - a varying-length character string up to n characters long
  - char(n) - a fixed-length character string of EXACTLY n characters
    - yes this PADS on the right with blanks if you try to store a value in such a column with fewer than n characters!
    - (style!) ONLY use for TRULY fixed-length-strings!

MORE intro to SQL and SQL*Plus

- here are a FEW of the more-common data types that the Oracle DBMS supports for columns in its implementation of SQL:
  - varchar2(n) - a varying-length character string up to n characters long
  - char(n) - a fixed-length character string of EXACTLY n characters
    - yes this PADS on the right with blanks if you try to store a value in such a column with fewer than n characters!
    - (style!) ONLY use for TRULY fixed-length-strings!
  - REMEMBER: in SQL, a string literal is surrounded by SINGLE QUOTES!!! (at least in column values...!)

MORE intro to SQL and SQL*Plus

- here are a FEW of the more-common data types that the Oracle DBMS supports for columns in its implementation of SQL:
  - varchar2(n) - a varying-length character string up to n characters long
  - char(n) - a fixed-length character string of EXACTLY n characters
    - yes this PADS on the right with blanks if you try to store a value in such a column with fewer than n characters!
    - (style!) ONLY use for TRULY fixed-length-strings!
  - REMEMBER: in SQL, a string literal is surrounded by SINGLE QUOTES!!! (at least in column values...!)
  - date - a true date type! even includes time!
    - we'll talk about lovely date-related functions later
    - in the meantime, here's the default way of typing a date literal when used in an insert statement:

MORE intro to SQL and SQL*Plus

- here are a FEW of the more-common data types that the Oracle DBMS supports for columns in its implementation of SQL:
  - date - a true date type! even includes time!
    - we'll talk about lovely date-related functions later
    - in the meantime, here's the default way of typing a date literal when used in an insert statement:

```
'DD-Mon-YYYY' or 'DD-Mon-YY'

'31-Aug-2021' or '31-Aug-21'
```

MORE intro to SQL and SQL*Plus

- here are a FEW of the more-common data types that the Oracle DBMS supports for columns in its implementation of SQL:
  - date - a true date type! even includes time!
    - we'll talk about lovely date-related functions later
    - in the meantime, here's the default way of typing a date literal when used in an insert statement:
    - fun fact: Oracle SQL also has a lovely no-argument function sysdate, that returns the current date...!

MORE intro to SQL and SQL*Plus

- here are a FEW of the more-common data types that the Oracle DBMS supports for columns in its implementation of SQL:
  - date - a true date type! even includes time!
  - decimal(p, q) - a decimal number of up to p total places, up to q of which are fractional
    - decimal(5, 2) - can hold values of up to 5 total places, 2 of which are fractional -- up to 999.99

MORE intro to SQL and SQL*Plus

- here are a FEW of the more-common data types that the Oracle DBMS supports for columns in its implementation of SQL:
  - date - a true date type! even includes time!
  - decimal(p, q) - a decimal number of up to p total places, up to q of which are fractional
  - integer - an integer in the range -2,147,482,648 to 2,147,483,647

MORE intro to SQL and SQL*Plus

- here are a FEW of the more-common data types that the Oracle DBMS supports for columns in its implementation of SQL:
  - date - a true date type! even includes time!
  - decimal(p, q) - a decimal number of up to p total places, up to q of which are fractional
  - integer - an integer in the range -2,147,482,648 to 2,147,483,647
    - don't try to specify the number of digits for an integer, Oracle SQL does not support that;

MORE intro to SQL and SQL*Plus

- here are a FEW of the more-common data types that the Oracle DBMS supports for columns in its implementation of SQL:
  - date - a true date type! even includes time!
  - decimal(p, q) - a decimal number of up to p total places, up to q of which are fractional
  - integer - an integer in the range -2,147,482,648 to 2,147,483,647
  - smallint - an integer in the range -32,768 to 32,767

MORE intro to SQL and SQL*Plus

- here are a FEW of the more-common data types that the Oracle DBMS supports for columns in its implementation of SQL:
  - date - a true date type! even includes time!
  - decimal(p, q) - a decimal number of up to p total places, up to q of which are fractional
  - integer - an integer in the range -2,147,482,648 to 2,147,483,647
  - smallint - an integer in the range -32,768 to 32,767
  - want to play it safe?
  - number - can store fixed and floating point numbers, virtually any magnitude, are guaranteed portable among different system running Oracle, up to 38 digits of precision...!

MORE intro to SQL and SQL*Plus

- A few starting word about PRIMARY keys
  - COURSE STYLE - every table should have an explicitly-defined primary key
  - a primary key is a SET of one or more columns whose values UNIQUELY determine a row in a table (no two rows can have the same value(s) for that set of columns!)
  - many/most DBMS's will ENFORCE this, and refuse to allow a row to be inserted that has the same primary key values as an existing row!

Question:

What is the primary purpose of a foreign key in a CREATE TABLE statement?

1) To uniquely identify each row in the table
2) To establish a relationship between a column in one table and a column in another table
3) To execute the previous SQL command
4) To display a message on the screen

MORE intro to SQL and SQL*Plus

- A few starting words about FOREIGN keys
  - these are what we use to RELATE the values in one table to the values in another table -- why we can say a database is made up of INTERRELATED tables
  - a foreign key is a set of one or more columns that are in another table, and the intent is that those columns in that table and those in this table have the same "meaning“

```
foreign key(col_name, ...) references desired_table

...that means the columns (col_name, ...) in this table
   reference those in desired_table

foreign key(col_name, ...) references desired_table(col_name, ...)

...use this form if the referenced column has a different name
   (but make sure the contents have the same MEANING...)
```

MORE intro to SQL and SQL*Plus

- A few starting words about FOREIGN keys
  - these are what we use to RELATE the values in one table to the values in another table -- why we can say a database is made up of INTERRELATED tables
  - a foreign key is a set of one or more columns that are in another table, and the intent is that those columns in that table and those in this table have the same "meaning“
  - IMPORTANT Oracle DBMS note: the foreign key column(s) MUST be the primary key of the referenced table...
  - NOW a proper DBMS will CHECK and make SURE a row being added to a table with a foreign key has a value for the foreign key column or columns that MATCH at least one row in the referenced table;
  - once you have foreign keys, you can't drop a referenced table if another table references it -- UNLESS your drop table statement ends in ... cascade constraints;

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
-- and typing a forward slash redoes the previous SQL statement;

/

-- playing with some of the column types discussed today

drop table parts cascade constraints;

create table parts
(part_num           integer,
 part_name          varchar2(25),
 quantity_on_hand   smallint,
 price              decimal(6, 2),  -- max price allowed is 9999.99
 level_code         char(3),        -- Must be exactly 3 characters
 last_inspected     date,
 primary key        (part_num)
);



-- INSERT --                                          36,1          Bot
```

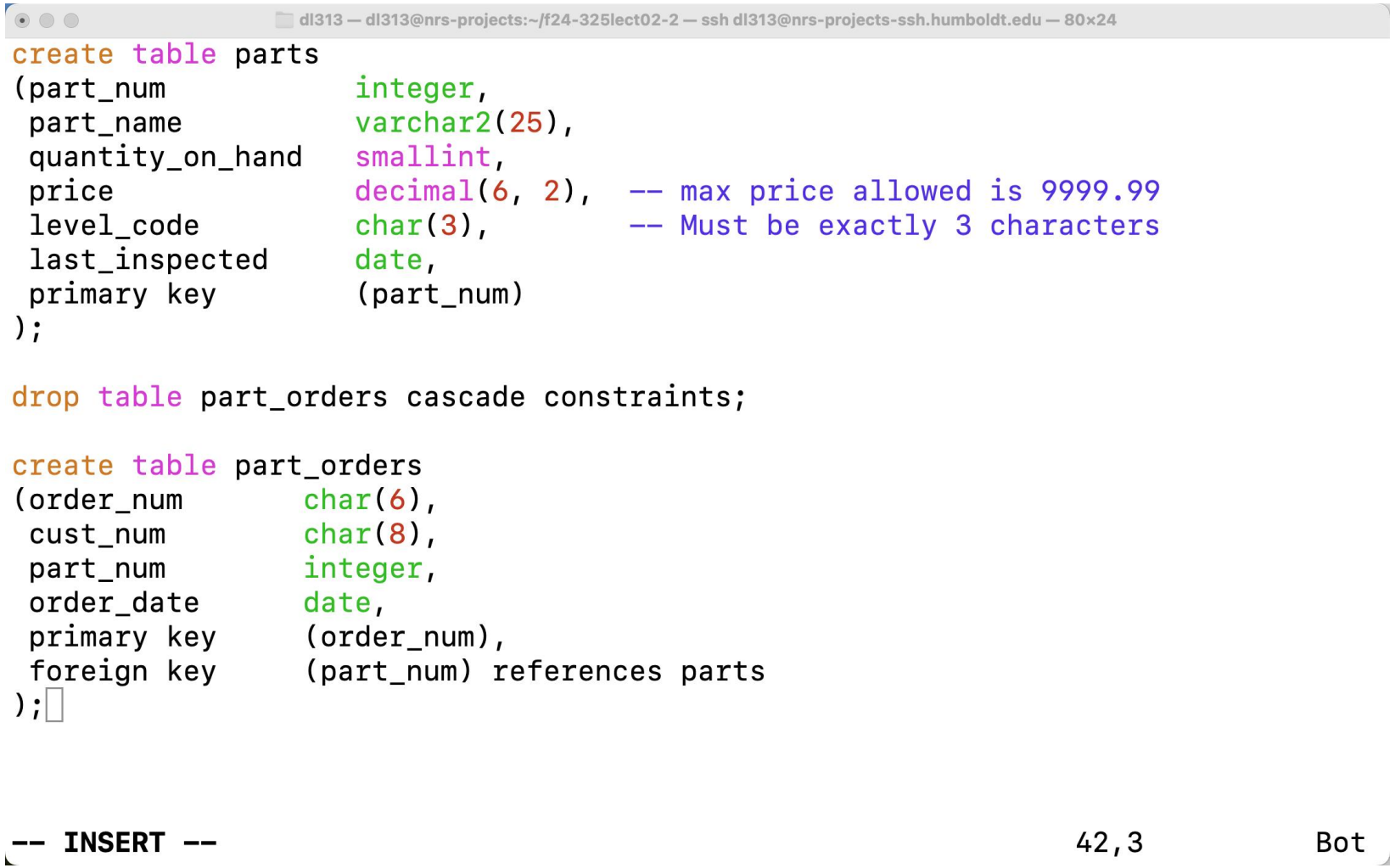
```
dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
       313 Dongcheng Li          black

SQL> start 325lect02-2.sql

Howdy there CS 325


STUDENT_ID STUDENT_NAME          STUDENT_SHIRTCOLOR
---------- -------------------- --------------------
       313 Dongcheng Li          black

demo forward slash

STUDENT_ID STUDENT_NAME          STUDENT_SHIRTCOLOR
---------- -------------------- --------------------
       313 Dongcheng Li          black


Table dropped.


Table created.

SQL>
```

MORE intro to SQL and SQL*Plus

- IMPORTANT NOTE!
  - do NOT put a blank line *within* a SQL statement!!!
  - (please put them BETWEEN SQL statements, but not WITHIN them)
    - sqlplus treats a blank line as if you are ending the current SQL statement...!

MORE intro to SQL and SQL*Plus

- SQL*Plus command: describe
  - use to describe the basic structure of the specified table
    - describe desired_table_name


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
create table parts
(part_num integer,
part_name varchar2(25),
quantity_on_hand smallint,
price decimal(6, 2), -- max price allowed is 9999.99
level_code char(3), -- Must be exactly 3 characters
last_inspected date,
primary key (part_num)
);
drop table part_orders cascade constraints;
create table part_orders
(order_num char(6),
cust_num char(8),
part_num integer,
order_date date,
primary key (order_num),
foreign key (part_num) references parts
);
-- INSERT --
42,3
Bot

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
);

drop table part_orders cascade constraints;

create table part_orders
(order_num       char(6),
 cust_num        char(8),
 part_num        integer,
 order_date      date,
 primary key     (order_num),
 foreign key     (part_num) references parts
);

prompt *** describe parts: ***

describe parts

prompt *** describe part_orders: ***

describe part_orders
```

-- INSERT -- 50,21 Bot

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
Table dropped.


Table created.

*** describe parts: ***
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PART_NUM                                  NOT NULL NUMBER(38)
 PART_NAME                                          VARCHAR2(25)
 QUANTITY_ON_HAND                                   NUMBER(38)
 PRICE                                              NUMBER(6,2)
 LEVEL_CODE                                         CHAR(3)
 LAST_INSPECTED                                     DATE

*** describe part_orders: ***
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_NUM                                 NOT NULL CHAR(6)
 CUST_NUM                                           CHAR(8)
 PART_NUM                                           NUMBER(38)
 ORDER_DATE                                         DATE

SQL>
```

MORE intro to SQL and SQL*Plus

- SQL*Plus command: describe
  - use to describe the basic structure of the specified table
    - describe desired_table_name
- SQL insert command
  - you can only insert one row per insert command
  - basic SQL insert syntax #1:
    - you MUST give a value for every column, in the order the columns appear in the create table statement

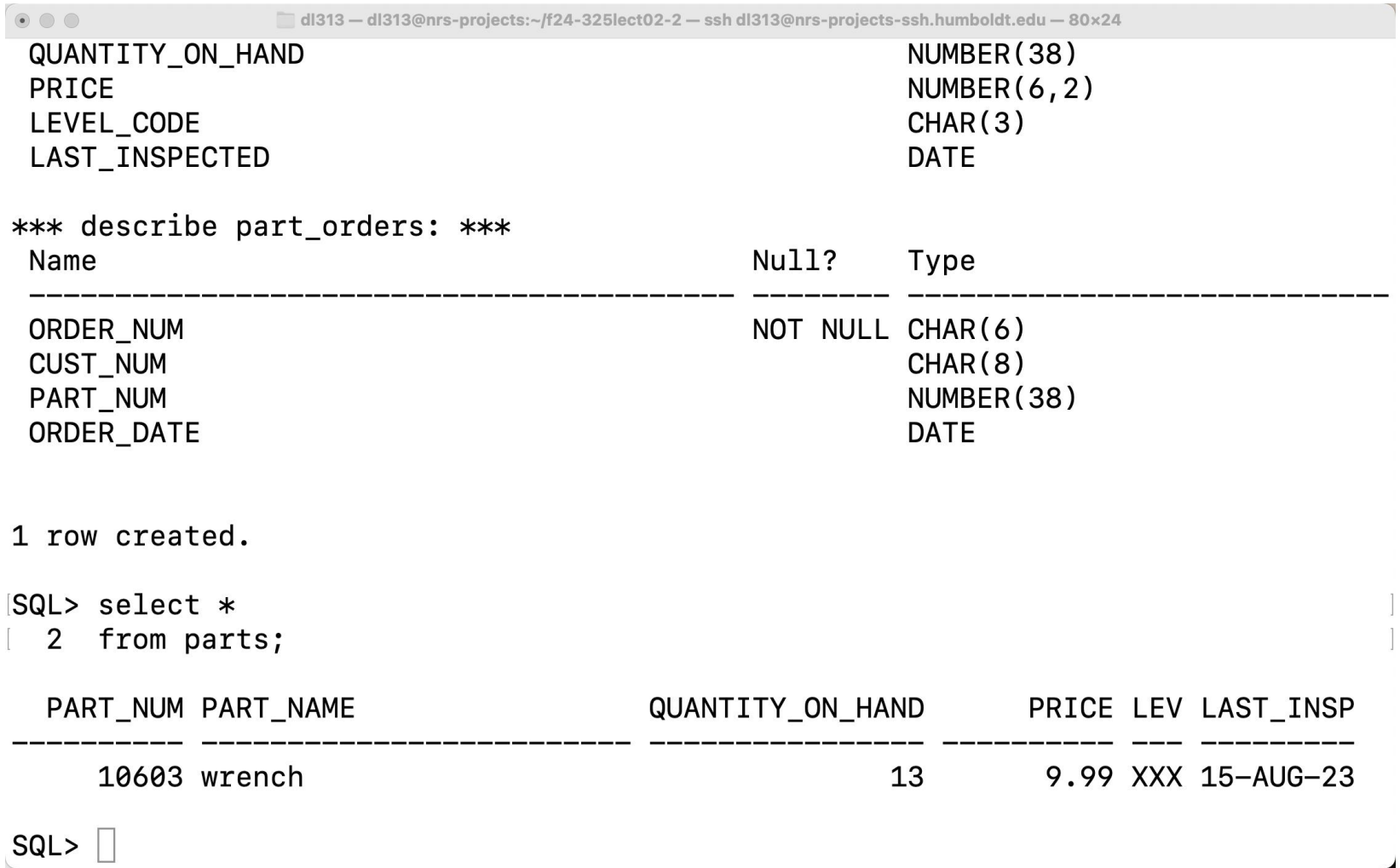
```
insert into desired_table
values
(col1_val, col2_val, ...);
```

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
 cust_num        char(8),
 part_num        integer,
 order_date      date,
 primary key     (order_num),
 foreign key     (part_num) references parts
);

prompt *** describe parts: ***

describe parts

prompt *** describe part_orders: ***

describe part_orders

-- let's add a few parts

insert into parts
values
(10603, 'wrench', 13, 9.99, 'XXX', '15-Aug-2023');

"325lect02-2.sql" 59L, 1166C written                          57,0-1        Bot
```

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
Table created.

*** describe parts: ***
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PART_NUM                                  NOT NULL NUMBER(38)
 PART_NAME                                          VARCHAR2(25)
 QUANTITY_ON_HAND                                   NUMBER(38)
 PRICE                                              NUMBER(6,2)
 LEVEL_CODE                                         CHAR(3)
 LAST_INSPECTED                                     DATE

*** describe part_orders: ***
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_NUM                                 NOT NULL CHAR(6)
 CUST_NUM                                           CHAR(8)
 PART_NUM                                           NUMBER(38)
 ORDER_DATE                                         DATE


1 row created.

SQL>
```


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
QUANTITY_ON_HAND NUMBER(38)
PRICE NUMBER(6,2)
LEVEL_CODE CHAR(3)
LAST_INSPECTED DATE
*** describe part_orders: ***
Name Null? Type
ORDER_NUM NOT NULL CHAR(6)
CUST_NUM CHAR(8)
PART_NUM NUMBER(38)
ORDER_DATE DATE
1 row created.
SQL> select *
2 from parts;
PART_NUM PART_NAME QUANTITY_ON_HAND PRICE LEV LAST_INSP
10603 wrench 13 9.99 XXX 15-AUG-23
SQL>

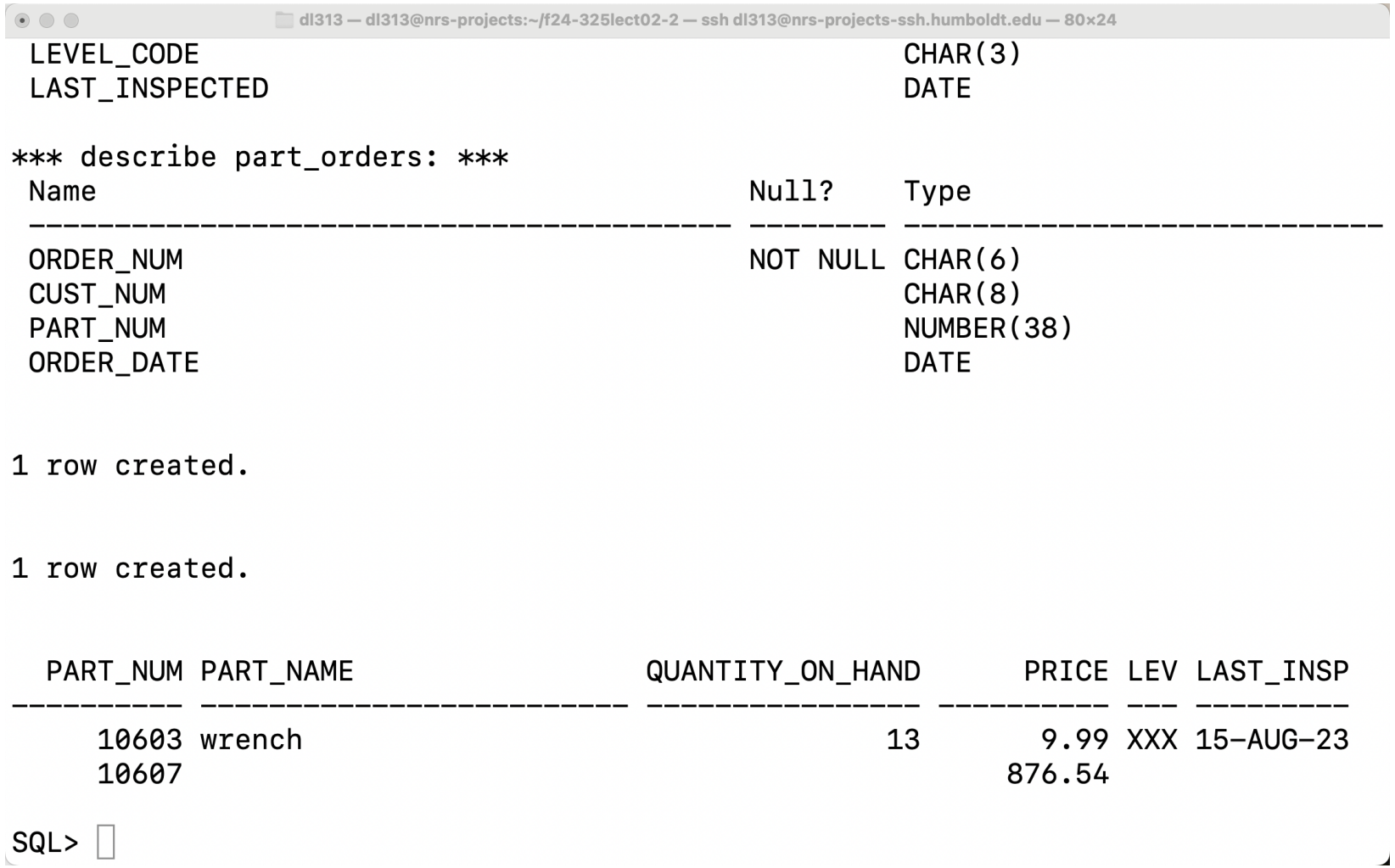
MORE intro to SQL and SQL*Plus

- SQL insert command
  - you can only insert one row per insert command
  - basic SQL insert syntax #2:
    - you don't have to give a value of every column!
    - (any you don't specify will be considered NULL, the special thing meaning empty, lack of a value! unless you specified a default value...
    - you need to specify the columns you ARE setting values to, and then give the values in that order:

```
insert into desired_table(first_col_to_set, second_col_to_set, ...)
values
(first_col_to_set_val, second_col_to_set_val, ...);
```

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
prompt *** describe parts: ***

describe parts

prompt *** describe part_orders: ***

describe part_orders

-- let's add a few parts

insert into parts
values
(10603, 'wrench', 13, 9.99, 'XXX', '15-Aug-2023');

insert into parts(price, part_num)
values
(876.54, 10607);

select *
from parts;

-- INSERT --                                    63,12         Bot
```


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
LEVEL_CODE CHAR(3)
LAST_INSPECTED DATE
*** describe part_orders: ***
Name Null? Type
ORDER_NUM NOT NULL CHAR(6)
CUST_NUM CHAR(8)
PART_NUM NUMBER(38)
ORDER_DATE DATE
1 row created.
1 row created.
PART_NUM PART_NAME QUANTITY_ON_HAND PRICE LEV LAST_INSP
10603 wrench 13 9.99 XXX 15-AUG-23
10607 876.54
SQL>

MORE intro to SQL and SQL*Plus

- a FEW additional CONSTRAINTS, or limitations, you can make when you declare a column in a table
  - not null
    - you want this column to ALWAYS have a value -- don’t let it be null (don't let it be empty)

```
create table blah
(...
 maxpoints    integer not null,
```

MORE intro to SQL and SQL*Plus

- a FEW additional CONSTRAINTS, or limitations, you can make when you declare a column in a table
  - Unique
    - you want a (non-primary-key) attribute to also be unique, not-repeated in any other row

MORE intro to SQL and SQL*Plus

- a FEW additional CONSTRAINTS, or limitations, you can make when you declare a column in a table
  - default desired_def_val
    - if a row is inserted that does not explicitly specify a value for this column, make that column have value desired_def_value

```
create table blah
(...
 quantity    integer default 1,
```

MORE intro to SQL and SQL*Plus

- a FEW additional CONSTRAINTS, or limitations, you can make when you declare a column in a table
  - check clause
  - check(boolean_expr)
    - only allow a value in this column for which boolean_expr is true

```
create table blah
(...
 opening_bid decimal(4, 2) check(opening_bid > 0 AND opening_bid < 50),
```

MORE intro to SQL and SQL*Plus

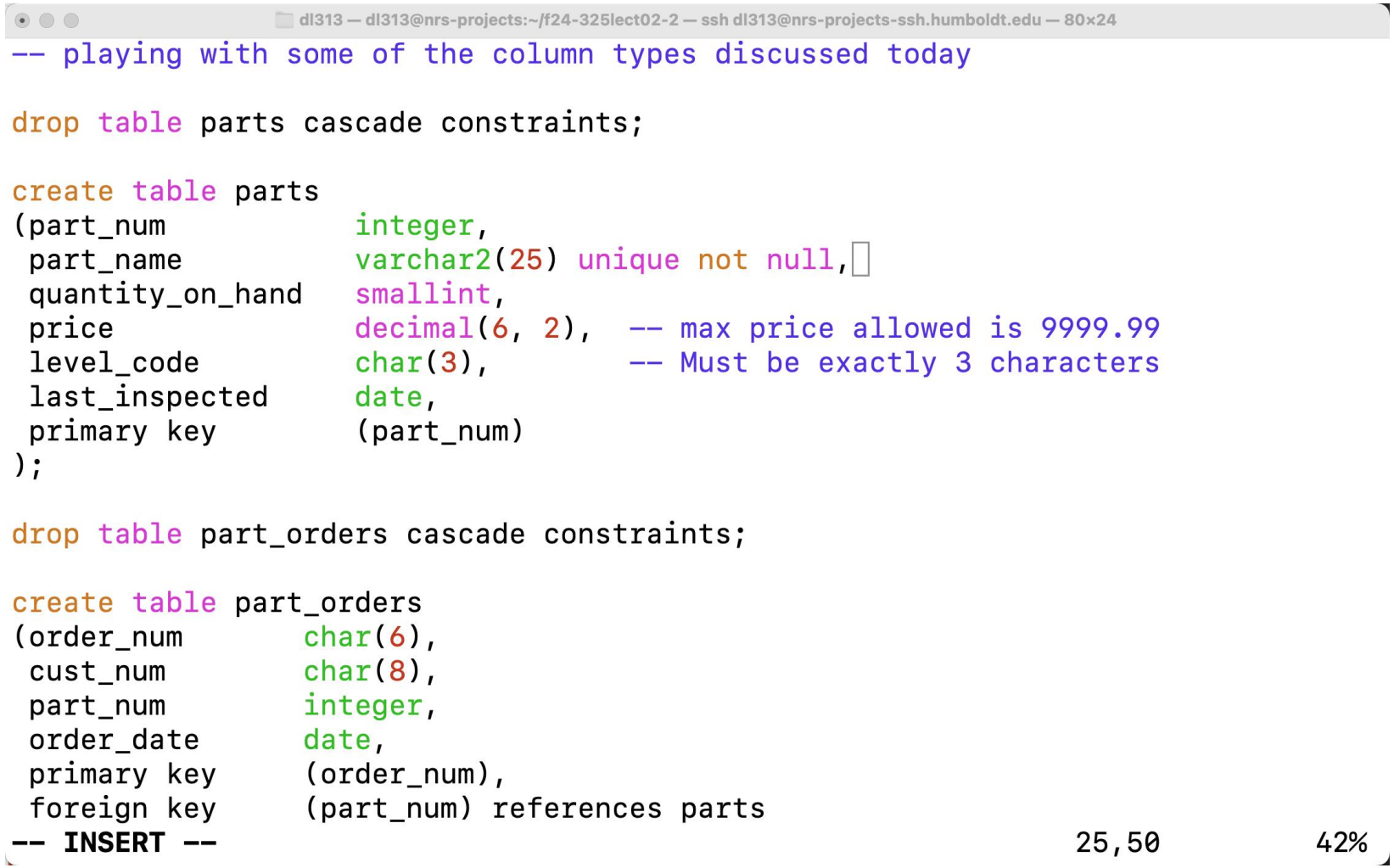
- a FEW additional CONSTRAINTS, or limitations, you can make when you declare a column in a table
  - Oracle SQL supports an IN operator!
    - val IN set …is true if val is a member of set

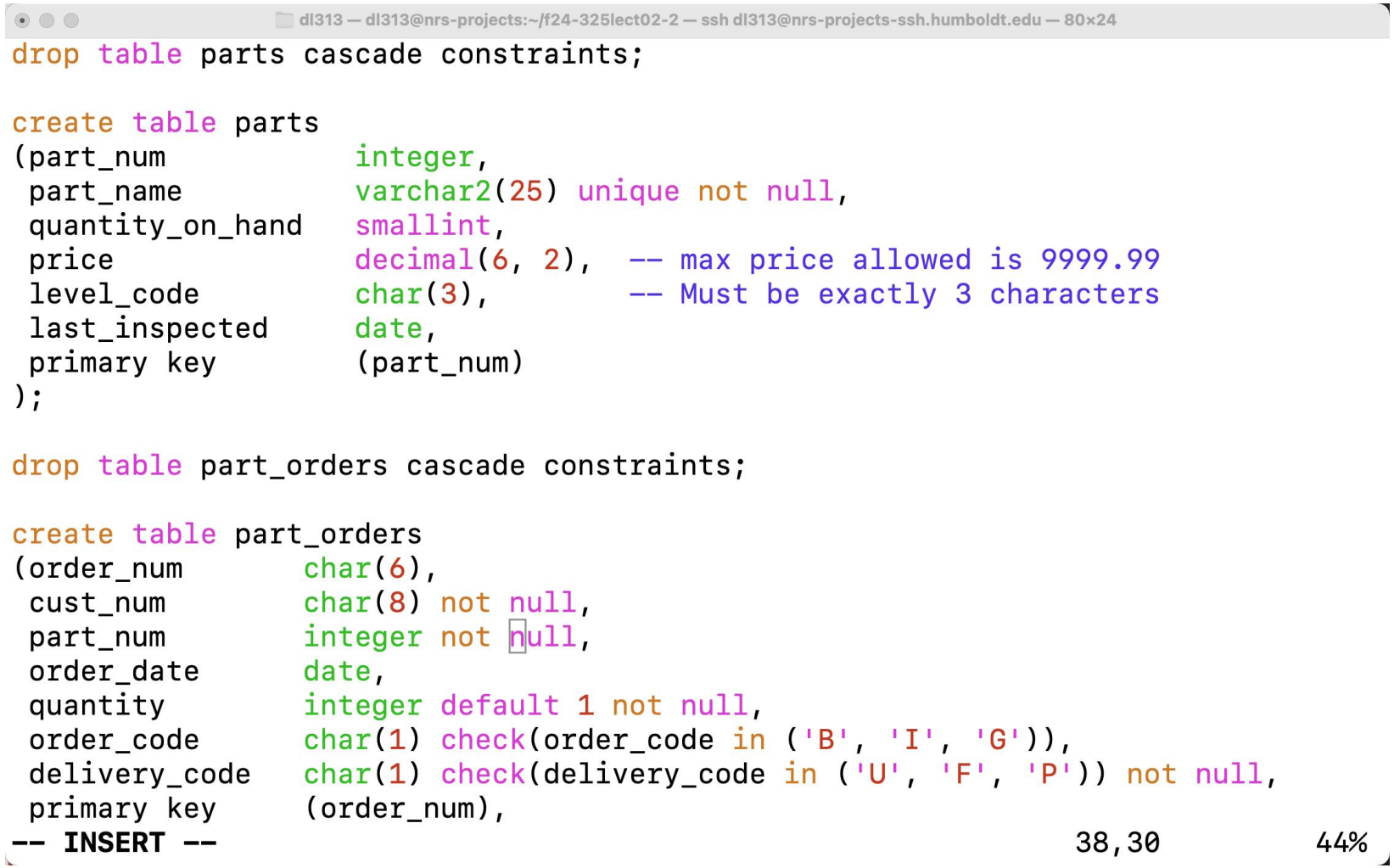
```
create table blah
(...
 car_color varchar2(5) check(car_color IN ('red', 'green',
'white')),
```

MORE intro to SQL and SQL*Plus

- a FEW additional CONSTRAINTS, or limitations, you can make when you declare a column in a table
  - Oracle SQL supports a BETWEEN operator!
    - val1 BETWEEN val2 AND val3 …is true if val1 >= val2 and val1 <= val3

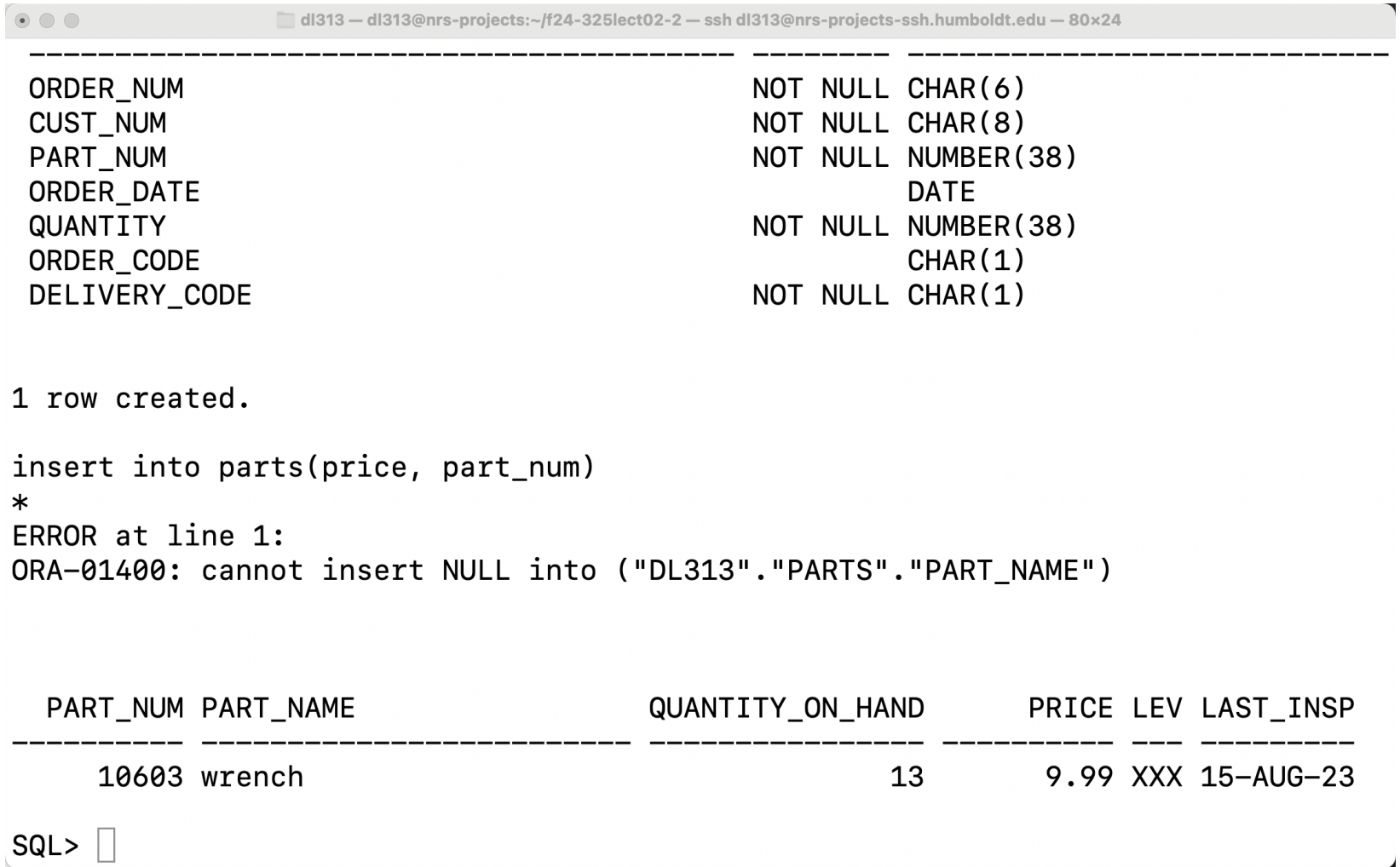
```
create table blah
(...
 quiz_grade decimal(5, 2) check(quiz_grade between 0.00 and
100.00),
```


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
-- playing with some of the column types discussed today
drop table parts cascade constraints;
create table parts
(part_num integer,
part_name varchar2(25) unique not null,
quantity_on_hand smallint,
price decimal(6, 2), -- max price allowed is 9999.99
level_code char(3), -- Must be exactly 3 characters
last_inspected date,
primary key (part_num)
);
drop table part_orders cascade constraints;
create table part_orders
(order_num char(6),
cust_num char(8),
part_num integer,
order_date date,
primary key (order_num),
foreign key (part_num) references parts
-- INSERT -- 25,50 42%


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
drop table parts cascade constraints;
create table parts
(part_num integer,
part_name varchar2(25) unique not null,
quantity_on_hand smallint,
price decimal(6, 2), -- max price allowed is 9999.99
level_code char(3), -- Must be exactly 3 characters
last_inspected date,
primary key (part_num)
);
drop table part_orders cascade constraints;
create table part_orders
(order_num char(6),
cust_num char(8) not null,
part_num integer not null,
order_date date,
quantity integer default 1 not null,
order_code char(1) check(order_code in ('B', 'I', 'G')),
delivery_code char(1) check(delivery_code in ('U', 'F', 'P')) not null,
primary key (order_num),
-- INSERT --
38,30
44%

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
prompt *** describe parts: ***

describe parts

prompt *** describe part_orders: ***

describe part_orders

-- let's add a few parts

insert into parts
values
(10603, 'wrench', 13, 9.99, 'XXX', '15-Aug-2023');

-- we made part_name not null, but we are trying to insert a row with part_name
as null, lets run and see what will happen

insert into parts(price, part_num)
values
(876.54, 10607);

select *
-- INSERT --                                                    61,123        93%
```


dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
ORDER_NUM NOT NULL CHAR(6)
CUST_NUM NOT NULL CHAR(8)
PART_NUM NOT NULL NUMBER(38)
ORDER_DATE DATE
QUANTITY NOT NULL NUMBER(38)
ORDER_CODE CHAR(1)
DELIVERY_CODE NOT NULL CHAR(1)
1 row created.
insert into parts(price, part_num)
*
ERROR at line 1:
ORA-01400: cannot insert NULL into ("DL313"."PARTS"."PART_NAME")
PART_NUM PART_NAME QUANTITY_ON_HAND PRICE LEV LAST_INSP
10603 wrench 13 9.99 XXX 15-AUG-23
SQL>

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
prompt *** describe parts: ***

describe parts

prompt *** describe part_orders: ***

describe part_orders

-- let's add a few parts

insert into parts
values
(10603, 'wrench', 13, 9.99, 'XXX', '15-Aug-2023');

insert into parts(price, part_name, part_num)
values
(876.54, 'bubble gum', 10607);

select *
from parts;


"325lect02-2.sql" 68L, 1479C written                          63,30         Bot
```

dl313 — dl313@nrs-projects:~/f24-325lect02-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
*** describe part_orders: ***
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORDER_NUM                                 NOT NULL CHAR(6)
 CUST_NUM                                  NOT NULL CHAR(8)
 PART_NUM                                  NOT NULL NUMBER(38)
 ORDER_DATE                                         DATE
 QUANTITY                                  NOT NULL NUMBER(38)
 ORDER_CODE                                         CHAR(1)
 DELIVERY_CODE                             NOT NULL CHAR(1)


1 row created.


1 row created.


  PART_NUM PART_NAME                 QUANTITY_ON_HAND      PRICE LEV LAST_INSP
---------- ------------------------- ---------------- ---------- --- ---------
     10603 wrench                                 13       9.99 XXX 15-AUG-23
     10607 bubble gum                                     876.54

SQL>
```