Create a publisher table (pub\_id, pub\_name). Create a book table (isbn, bk\_name) (identify primary key with constraint name). Then do the following:

* Add numberOfPages column, then place a check constraint on the it (should not be more than 200 pages). Add copy\_right date column. Come up with a primary key for publisher table. Create a foreign key. Insert records to verify use mm dd yyyy format (for one of the inserts, identify the columns). Display the date info (including the hours and minutes in military time). Create an index on book table. Give a listing of all the constraints, then all the indexes, then all the tables created. Disable foreign key. Drop check constraint. Truncate table.

CREATE TABLE publisher (

pub\_id NUMBER,

pub\_name VARCHAR2(20)

);

CREATE TABLE book (

isbn NUMBER CONSTRAINT bk\_pk PRIMARY KEY,

bk\_name VARCHAR2(20)

);

* ALTER TABLE book ADD numberOfPages NUMBER;
* ALTER TABLE book ADD CONSTRAINT pages\_ck CHECK (numberOfPages < 200); *or* ALTER TABLE book MODIFY numberOfPages CONSTRAINT pages\_ck CHECK (numberOfPages < 200);
* ALTER TABLE book ADD copy\_right DATE;
* ALTER TABLE publisher ADD CONSTRAINT pub\_pk PRIMARY KEY (pub\_id); *or* ALTER TABLE publisher MODIFY pub\_id CONSTRAINT pub\_pk PRIMARY KEY;
* ALTER TABLE book ADD pub\_id NUMBER; ALTER TABLE book ADD CONSTRAINT bk\_fk FOREIGN KEY (pub\_id) REFERENCES publisher; or ALTER TABLE book MODIFY pub\_id CONSTRAINT bk\_fk REFERENCES publisher; (this whole thing can be combined to one line as follows:)
* OR ALTER TABLE book ADD pub\_id CONSTRAINT bk\_fk REFERENCES publisher;
* INSERT INTO publisher VALUES (1, ‘pub’); INSERT INTO book VALUES (111, ‘Bad book’, to\_date(‘01 01 1991’, ‘mm dd yyyy’), 190, 1); (sequence is as follows: isbn, bkname, copy\_right, numberOfPages, pub\_id)
* SELECT to\_char (copy\_right, ‘mm //dd//yy hh24:mi’) FROM book;
* CREATE INDEX inxBook ON book (bkname);
* SELECT constraint\_name, r\_constraint\_name, table\_name, constraint\_type, status, search\_condition FROM user\_constraints WHERE table\_name IN (‘PUBLISHER’, ‘BOOK’);
* SELECT table\_name, index\_name FROM user\_indexes;
* SELECT table\_name FROM user\_tables;
* ALTER TABLE book DISABLE CONSTRAINT bk\_fk;
* ALTER TABLE DROP CONSTRAINT bk\_ck;
* TRUNCATE TABLE book;

***Example 3.1a (Create table)***

CREATE TABLE hello (name, id);

***Example 3.1b (DESC command)***

DESC provides information about the columns in a table. Semicolon not required at the end. DESC patient;

***Example 3.1c (View contents of table)***

SELECT \* FROM patient;

***Example 3.1d (Drop table command)***

Moves table to recycle bin. All data is erased

Drop TABLE patient;

***Example 3.1e (Flashback)***

*A dropped table can be recovered using Flashback*

FLASHBACK TABLE patient TO BEFORE DROP;

***Example 3.1f (Recyclebin)***

Purging from the recyclebin

PURGE RECYCLEBIN;

***Example 3.1g (Not NULL and default constraints)***

Create a table with a NOT NULL and DEFAULT constraint. If no value is entered for a column, the value is considered NULL, indicating an absence of data.

CREATE TABLE patient (pat\_id NUMBER NOT NULL, fname VARCHAR..);

***Example 3.1h (Insert statement)***

Insert two rows of data into the table. Notice the data that is being inserted into the table is case-sensitive, but the syntax is not. Will insert values into top table.

INSERT INTO patient VALUES (11,'John', 'Smith', 'm','01-FEB-1970', 55000

***Example 3.1k (Delete)***

The table will still exist but its contents will be deleted. The data can still be recovered if an implicit or explicit commit has not been implemented. Delete can be applied to all or specific rows in a table. Notice no asterisk

DELETE FROM patient;

***Example 3.1l (Truncate)***

The table will still exist but its contents will be deleted. The data cannot be recovered. It is a lot faster than delete. Unlike delete, it is applied to all the rows in the table. TRUNCATE TABLE patient;

**3.2 Adding columns using ALTER command**

Used for structural changes to a table. Example, you might need to add a column, delete, or change column’s size. Changes are made with ALTER TABLE.

ALTER TABLE tablename ADD|MODIFY|DROP columnname (definition); Using an ADD clause with the ALTER TABLE command allows a user to add a new column to a table.

***Example 3.2a (Additional column)*** --This statement adds marital\_status as a column to the table patient. The datatype of this new --column is CHAR. ALTER TABLE patient ADD marital\_status CHAR;

***Example 3.2b (Adding multiple columns)***

Adding multiple columns at the same type --For multiple columns, the syntax looks like the create table statement.

ALTER TABLE patient ADD (

Height NUMBER,

Weight NUMBER

***3.3 Modifying using ALTER command***

*To change an existing column’s definition, we use MODIFY. With the ALTER TABLE:*

* Changing the column size (increase or decrease)
* Changing the datatype (such as VARCHAR2 to CHAR)
* Changing or adding the default value of a column (such as DEFAULT SYSDATE)

*Rules when modifying existing columns:*

* A column must be as wide as the data fields it already contains
* If a NUMBER column already contains data, you can’t decrease the column’s precision or scale.
* Changing the default value of a column doesn’t change the values of data already in the table

***Example 3.3a (Modify)***

ALTER TABLE patient MODIFY fname VARCHAR2(20);

***Example 3.4b (Dropping columns)***

Getting rid of multiple columns.

ALTER TABLE patient DROP (annual\_salary, marital\_status);

Example 3.4c (Set unused)

Setting columns to unused and then dropping them (Cannot be rolled back). --Cannot recover the data

ALTER TABLE patient SET UNUSED (DOB, gender);

DESC patient;

--Cannot recover the data

ALTER TABLE patient DROP UNUSED COLUMNS;

***Example 3.5a (Renaming a column)***

Renames a column from fname to first\_name. Only one column can be renamed at a time. ALTER TABLE patient RENAME COLUMN fname TO first\_name;

***3.6 Constraints***

constraint at the column level means the constraint’s definition is included as part of the column definition, similar to assigning a default value to a column

Creating a constraint at the table level means the constraint’s definition is separate from the column definition.

The main difference in the syntax of a column- level constraint and a table- level constraint is that you provide column names for the table- level constraint at the end of the constraint definition inside parentheses, instead of at the beginning of the constraint definition.

***3.7 Primary key constraint***

***Example 3.7a*** (Primary key at the column level) The constraint makes certain the columns identified as the table’s primary key are unique and do not contain NULL values.

DROP TABLE patient;

CREATE TABLE Patient (

Patient\_id NUMBER PRIMARY KEY,

Fname VARCHAR2(20),

Lname VARCHAR2(20).

INSERT INTO patient VALUES (11,'John', 'Smith');

***Example 3.7b (User\_constraints\_table)***

Examining the system table user\_constraints --Constraint\_type (p)

stands for primary key. The constraint name is generated by the

system. PATIENT must be typed as upper-case

SELECT table\_name, constraint\_name, constraint\_type FROM user\_constraints WHERE table\_name='PATIENT'

***Example 3.7d (Primary key at the table level)***

Creating a primary key constraint at the table level. DROP TABLE

patient;

--Use the CONSTRAINT keyword when giving a name to a constraint.

--The constraint is created at the table level after all column

definitions. CREATE TABLE Patient (

Patient\_id NUMBER,

Fname VARCHAR2(20),

Lname VARCHAR2(20),

CONSTRAINT patient\_patient\_id\_pk PRIMARY KEY (patient\_id);

DROP TABLE patient;

***Example 3.7e (Composite primary key)***

This example creates a composite primary key, with the

assumption that the combination of fname and lname

are unique. Composite key cannot be created at the column level. DROP TABLE patient;

--Invalid example: Cannot have two primary keys.

--The syntax for composite primary key requires a table level

syntax.

CREATE TABLE Patient(

Patient\_id NUMBER,

Fname VARCHAR2(20) PRIMARY KEY,

Lname VARCHAR2(20) PRIMARY KEY

--Here is the syntax for a composite primary key with a constraint name.

--Notice the constraint keyword is used to give a name to the constraint.

CREATE TABLE Patient(

Patient\_id NUMBER,

Fname VARCHAR2(20),

Lname VARCHAR2(20),

CONSTRAINT patient\_fname\_lname\_pk

PRIMARY KEY (Fname, Lname)

***Example 3.7f (Using Alter table command)***

Creating a primary key using the alter command DROP TABLE

patient; CREATE TABLE Patient(

Patient\_id NUMBER,

Fname VARCHAR2(20),

Lname VARCHAR2(20)

--Can create a primary key using alter table statement.

ALTER TABLE patient ADD PRIMARY KEY(patient\_id);

3.8 Unique constraint

***Example 3.8a (Unique key at the column level)***

Creating a unique key constraint at the

column level DROP TABLE patient;

--Use the UNIQUE keyword to create a

candidate or unique key.

--A table can have many unique keys. Also,

unlike a primary key which cannot-be NULL,

a unique key

--column can have NULLs.

CREATE TABLE Patient

(

Patient\_id NUMBER UNIQUE,

Fname VARCHAR2(20),

Lname VARCHAR2(20)

***Example 3.8d (Unique key at the table level)***

Creating a unique key constraint at the table level DROP TABLE patient;

--Unique key constraint can also be created at the table level just like a primary key.

--The constraint keyword is used to give a name to the unique key.

CREATE TABLE Patient (

Patient\_id NUMBER,

Fname VARCHAR2(20),

Lname VARCHAR2(20),

CONSTRAINT patient\_patient\_id\_uk UNIQUE (patient\_id)

***Example 3.8e (Composite unique key)***

This example creates a composite unique key

with the assumption that the combination of

fname and lname must be unique. Composite key cannot be created at the column level.

DROP TABLE patient;

--Unlike a primary key, a table can have multiple unique keys. This is not a composite, unique key. Fname and

--lname are unique by themselves. The system generates a name.

CREATE TABLE Patient (

Patient\_id NUMBER,

Fname VARCHAR2(20) UNIQUE,

Lname VARCHAR2(20) UNIQUE

--Composite unique key has the same basic syntax as a composite primary key. CREATE TABLE Patient (

Patient\_id NUMBER,

Fname VARCHAR2(20),

Lname VARCHAR2(20),

CONSTRAINT patient\_fname\_lname\_uk UNIQUE (Fname, Lname)

***Example 3.8f (Creating unique with alter table***

***command)*** CREATE TABLE Patient (

Patient\_id NUMBER,

Fname VARCHAR2(20),

Lname VARCHAR2(20)

);

ALTER TABLE patient ADD CONSTRAINT patient\_patient\_id\_uk UNIQUE (patient\_id);

--Can use MODIFY for a unqique constraint without

the constraint keyword. ALTER TABLE patient MODIFY patient\_id UNIQUE;

--Use the constraint keyword to give a name to the constraint.

ALTER TABLE patient MODIFY patient\_id CONSTRAINT

patient\_patient\_id\_uk UNIQUE;

***Example 3.8g (Using alter to create composite***

***unique key)***

|  |
| --- |
| This example creates a composite unique key using the alter command.  DROP TABLE patient;  CREATE TABLE Patient(  Patient\_id NUMBER,  Fname VARCHAR2(20),  Lname VARCHAR2(20)  --Creates a composite unique key after the table has been created.  ALTER TABLE patient ADD UNIQUE(fname, lname); |

***Example 3.9a (Column v. table level check***

***constraint)*** Column level versus table level without a constraint name DROP TABLE patient;

--Use the Check keyword to create a check constraint. The name for this constraint will be --generated by the system. This constraint is created at the column level because the comma --appears after the column name, datatype and the actual check constraint. CREATE TABLE Patient(

Patient\_id NUMBER,

Height NUMBER CHECK (height>10),

Fname VARCHAR2(20),

Lname VARCHAR2(20)

***Example 3.9c (Check constraint table level vs column level)***

--The check constraint, created at the column level, is given a name using the CONSTRAINT keyword. CREATE TABLE Patient (

Patient\_id NUMBER,

Height NUMBER CONSTRAINT patient\_height\_ck CHECK height>10)

--This check constraint is created at the table level. CREATE TABLE Patient (

Patient\_id NUMBER,

Height NUMBER,

Fname VARCHAR2(20), Lname VARCHAR2(20),

CONSTRAINT patient\_height\_ck CHECK (height>10) );

***Example 3.9d (Using alter table to create check***

***constraint)***

Creating a check constraint using the alter statement with and without a constraint name DROP TABLE patient;

CREATE TABLE Patient(

Patient\_id NUMBER,

Height NUMBER,

Fname VARCHAR2(20),

Lname VARCHAR2(20)

);

ALTER TABLE patient MODIFY height CONSTRAINT patient\_height\_ck CHECK(height>10);

***Example 3.10a (Not NULL constraint)***

NULL constraint at table level where constraint does not have a name CREATE TABLE Patient ( Patient\_id NUMBER NOT NULL, Height NUMBER);

NULL constraint with name --Like any other constraint, you can give this constraint a name. CREATE TABLE Patient ( Patient\_id NUMBER CONSTRAINT patient\_patient\_id\_nn NOT NULL);

***Example 3.10a (Using modify command)***

ALTER TABLE patient MODIFY patient\_id NUBMER NOT NULL; OR

ALTER TABLE patient MODIFY patient\_id NUMBER CONSTRAINT patient\_pat\_id\_nn NOT NULL;

**3.11 Foreign Key constraints**

***Example 3.11a (References keyword)***

--To create a foreign key, go to the many table and use the references keyword to point to the --column on the one table. The column on the one side should be a primary key.Use constraint --keyword to give the foreign key a name.

CREATE TABLE patient\_disease (

Patient\_id NUMBER REFERENCES patient (patient\_id),

Disease\_id NUMBER CONSTRAINT dis\_id\_fk REFERENCES disease, PRIMARY KEY (patient\_id, disease\_id) );

***Example 3.11c (Foreign keys and delete statements)***

--Note the order of deletes. Must delete from patient\_disease to get rid -of the association. DELETE FROM patient\_disease;

***Example 3.11d (Truncate vs Delete)***

You cannot use TRUNCATE TABLE on a table referenced by a FOREIGN KEY constraint; instead, use DELETE statement without a WHERE clause. You have to get rid of the foreign key constraint(s) if you want to truncate the parent table.

***Example 3.11e (Delete cascade)***

Delete cascade option allows for a record in the parent table to be deleted even if it is being referenced in the child table. Using this option will remove all the corresponding records in the child table in order to maintain referential integrity

***Example 3.11g (Foreign key at table level)***

Notice the syntax FOREIGN KEY is used to refer to the column in the child table and the references syntax is used to refer to the column in the parent table

--Foreign key is being created at the table level. This requires the additional -(FOREIGN KEY) keyword. --Since all the columns have been defined in the child table already, we need to identify the --column that will be the foreign key in the child table and then we use the REFERENECES keyword --as in the previous syntax. Once again the CONSTRAINT keyword is needed to give it a name.

CREATE TABLE patient\_disease (

Patient\_id NUMBER,

Disease\_id NUMBER,

PRIMARY KEY (patient\_id, disease\_id),

FOREIGN KEY (patient\_id) REFERENCES patient ON DELETE CASCADE, CONSTRAINT patient\_disease\_disease\_id\_fk FOREIGN KEY (disease\_id) REFERENCES disease );

***Example 3.11h (using alter table to create Foreign key)***

ALTER TABLE patient\_disease ADD FOREIGN KEY(patient\_id) REFERENCES patient;

ALTER TABLE patient\_disease ADD CONSTRAINT patient\_disease\_disease\_id\_fk FOREIGN KEY(disease\_id) REFERENCES disease;

**3.12 Disabling/enabling/dropping constraints**

***Example 3.12a (disable/enable/drop constraints)***

ALTER TABLE patient DISABLE PRIMARY KEY;

--The status column conveys if the constraint is enabled or disabled SELECT constraint\_name, constraint\_type, status FROM user\_constraints WHERE table\_name='PATIENT';

--Constraints can also be dropped. Once dropped, there is no way to get them back. They have to be --recreated, which is different from disabled constraints.

ALTER TABLE patient DROP CONSTRAINT patient\_height\_ck; ALTER TABLE patient DROP PRIMARY KEY; ALTER TABLE patient DROP UNIQUE (fname,lname);

3.13 Indexes

An index can be created implicitly or explicitly. Oracle creates an index automatically when a PRIMARY KEY or UNIQUE constraint is created for a column.

***Example 3.13a (Indexes)***

CREATE INDEX patient\_address\_idx ON patient (address);

--The primary key and unique key information will be in the user\_constraints table. SELECT constraint\_name, constraint\_type FROM user\_constraints WHERE table\_name='PATIENT'; --The index that was created using create index command along with the primary key and unique --key indexes will be in this table. The name of the indexes for the primary key and unique will be --the same as the constraint names that appear in the user\_constraints table SELECT index\_name FROM user\_indexes WHERE table\_name='PATIENT';

***Example 3.13b (Alter Index command)***

--The name of the index is altered. ALTER INDEX patient\_address\_idx RENAME TO pat\_add\_idx;

***Example 3.13c (Drop index)***

DROP INDEX patient\_address\_idx;

**EXTRA NOTES**

**NULL insert**

INSERT INTO sometable VALUES (NULL, ‘col2’, ‘col3’); (no quotes around NULL)

**NUMBERS**

**NUMBER(5)🡪** (only accepts an integer and will round numbers on the first decimal)

NUMBER(5,2) 🡪 (5 digits long total, 2 of which are decimal digits. Will round extra decimals.)

NUMBER(5,3)🡪 (won’t allow ‘123’ because only 2 digits are allowed to the left of the decimal)

**DATE**

DD-MON-YY (19-APR-99), Month (Month spelled out), MON (3-letter abbreviation), MM (2-digit month), RM (Roman Numeral month), D (numerical day of the week), DD (numerical day of the month), DDD (numerical day of the year), DAY (named day of the week), DY (3-letter abbreviation for day), YYYY (4-digit year, YYY/YY/Y (numerical value for last 3, 2 or 1 digit of the year), YEAR (spelled out version of the year), BC/AD (Indicates BC or AD), SS (seconds), SSSS (seconds past midnight), MI (minutes), HH/HH12 (hours), HH24 (military time), AM/PM (indicates AM or PM)

**1A Create a student table that will hold the following data. Make sure you assign a primary key**

CREATE TABLE student (

Ssn VARCHAR2(20) PRIMARY KEY,

Lname VARCHAR2(20),

Fname VARCHAR2(20),

Age NUMBER,

Salary NUMBER,

Dob NUMBER

);

**1B) After the table has been created add a candidate key based on lname and fname.**

Note: Candidate and unique key are the same thing

ALTER TABLE student ADD CONSTRAINT student\_lname\_fname UNIQUE (Lname,Fname);

**1C) After the table has been created add a check constraint such that the age is greater than 10 but less than 50. Provide a name for the check constraint.**

ALTER TABLE student MODIFY age CHECK (AGE>=10 AND AGE<50);

**1D) After the table has been created add a column called address.**

ALTER TABLE STUDENT ADD Address varchar2(30);

**1E) After the table has been created, modify the dob column to be of datatype date and also not null**

ALTER TABLE student MODIFY Dob DATE NOT NULL;

**1F) Create a composite index on ssn and dob**

CREATE INDEX ssn\_dob\_idx ON Student (ssn,dob)

**1G) After the table has been created add a column called transferable with a not null constraint. Do not assign a name to the constraint**

ALTER TABLE student ADD Transeferable VARCHAR2(20) NOT NULL;

**1H) After the table has been created add a check constraint on the column transferable to allow only ‘y’,’Y’,’n’,’N’. Give the constraint a name.** ALTER TABLE student ADD CONSTRAINT col\_trans CHECK(Transeferable='y' AND Transeferable='Y' AND Transeferable='n' AND Transeferable='N');

ALTER TABLE student ADD CONSTRAINT col\_trans CHECK (transeferable IN(‘Y’,’y’,’N’,’n’));

**2A) Create a second table called class that will hold the following data. You decide what the data types are going to be.**

CREATE TABLE class (

code NUMBER,

descrip varchar2(20) UNIQUE NOT NULL

);

CREATE TABLE class (

code NUMBER,

descrip varchar2(20) NOT NULL,

UNIQUE (descript)

);

**2B) After the table has been created add the primary key. Give the constraint a name** ALTER TABLE class MODIFY Code Primary Key;

**2C) Create an index on class description**

CREATE INDEX class\_descp\_idx ON class (descrip);

**3A) Create a third table called student\_class. This table is an association table that contains information on the different class that the students are taking. You figure out what the columns should be. It should contain only two columns.**

CREATE TABLE student\_class (

ssn varchar2(20),

classCode NUMBER

)

**3B After the table has been created add the primary key constraint (Name the constraint)**

ALTER TABLE student\_class ADD CONSTRAINT student\_class\_pk PRIMARY KEY(ssn, classCode);

**3CAfter the table has been created add the foreign key constraint(s) (Name the constraint(s))**

**Examples:**