# Assignment 1 5%

Introduction to Oracle 12c SQL and DB concepts

## Due Date: October 5, 2017 midnight

**Part 1** Maintaining and redesigning an existing database (2 %)

Use your database created from the SQL Developer Tutorial Script (LAB 1). You must use your server account on BTACS database and perform all queries using your own account. This data base should have three tables.

books ( book\_id, title, author\_last\_name, author\_first\_name, rating)

patrons (patron\_id, last\_name, first\_name, street\_address, city\_state\_zip, location)

transactions (transaction\_id, patron\_id, book\_id, transaction\_date, transaction\_type)

Use the existing database to do the following:

1. **Adding new data (DML):**
   1. Add a new book to the BOOKS table: add **your favorite book** (add a high rating☺).

**INSERT INTO books VALUES ('BK001','Harry Potter', 'Rowling', 'J.K', 9);**

* 1. Borrow your favorite book. Use yourself as a patron (added in Lab1). Return your favorite book.

**Borrowing: INSERT INTO transactions VALUES (12, 100,'BK001', SYSDATE, 1);**

**Returning: INSERT INTO transactions VALUES (13, 100,'BK001', SYSDATE, 2);**

* 1. List **the last transaction** (all columns) for **your favorite book.**

**SELECT \* FROM transactions WHERE book\_id = 'BK001' AND transaction\_date = (SELECT MAX (transaction\_date) FROM transactions);** 

Print the SQL queries for a-c and the result for c.

1. **Adding new columns (DDL):**
   1. Write an SQL statement to add a new column DOB to the PATRONS table. Print the query.

**ALTER TABLE patrons ADD DOB DATE;**

* 1. Write an SQL statement to add LAST\_MODIFIED and MODIFIED\_BY columns to the PATRONS table. The LAST\_MODIFIED column will have the server date and time and the MODIFIED\_BY will have the USER name.

**ALTER TABLE patrons ADD (**

**LAST\_MODIFIED DATE DEFAULT (SYSDATE),**

**MODIFIED\_BY VARCHAR2 (30));**

Print the DDL statements for a and b.

1. **Changing data (DML):**
   1. Add DOB to an existing patron (yourself). Use the TO\_DATE function for the DOB.

**UPDATE patrons**

**SET dob = TO\_DATE ('05/10/1995','dd/mm/yyyy')**

**WHERE patron\_id = 100;**

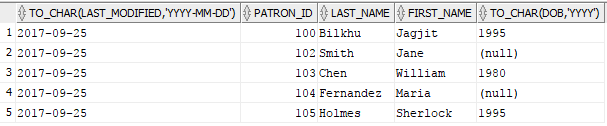
* 1. Add another patron with all data (including columns added in 2b).

**INSERT INTO patrons VALUES (105,'Holmes', 'Sherlock','221 B Baker Street', 'Kamloops BC V2B4N3', null, to\_date ('01/10/1995', 'dd/mm/yyyy'), default, 'JB');**

* 1. List the patrons (Last\_modified, patron\_id, last\_name, first\_name, DOB) whose data were added/modified **within last 10 days** (use SYSDATE as today’s date). Use ISO standard for LAST\_MODIFIED including local time. Use only the year for the DOB (YYYY). Sort the results by transaction date (newest transactions first).

**SELECT to\_char (last\_modified,'yyyy-mm-dd'), patron\_id, last\_name, first\_name, to\_char (DOB,'yyyy')**

**FROM PATRONS**

**WHERE LAST\_MODIFIED >= SYSDATE-10 ORDER BY LAST\_MODIFIED;** 

Print the SQL queries for a-c and the result for c.

**Part 2** Data Modeling for the TRU Sleep Clinic (2 %)

Use the following information/knowledge to prepare data requirements for the TRU Sleep Clinic.

TRU sleep clinic runs a special study on blood pressure (BP) and sleep disorders (specifically obstructive sleep apnea). The following data has to be kept for each patient: patient ID, gender, DOB, date/time of assessment, weight, height, BMI (calculated from weight and height), blood pressure: systolic and diastolic BP, antihypertensive medication (yes/no). The data for the weight, height, BP, and AH medication are recorded once a month. The weight and height should be stored using the International System of Units (SI standard/ metric).

1. Create a **Logical Data Model** (use Relational Model) for the above study: list the relations with attributes, PKs, and FKs. For each attribute provide a short description (meaning), domain, standard for units/representations. Patient ID is a unique integer number assigned by the Clinic Coordinator. Gender should allow for undeclared values (represented by ‘U’) and unknown (missing values) as NULL.

**PATIENT (patient\_id, gender, DOB, weight, height, BMI, systolic\_bp, diastolic BP, antihypertensive);**

|  |  |  |  |
| --- | --- | --- | --- |
| PATIENT | | | |
| ATTRIBUTES | **ATTRIBUTE NAME** | **MEANING** | **DOMAIN DEFINITION** |
| PATIENT\_ID | **Patient ID** | **Patient ID assigned by clinic Coordinator** | **Character: Unique for each patient** |
| GENDER | **Gender** | **Patients Gender:**  **Can hold values Male, Female, U and Null.**  **U if patient does not declare and Null if not recorded** | **Character: Specific Values: Male, Female, Undeclared Null** |
| DOB | **Date Of Birth** | **Date of birth of patient** | **Date: range from (1st January 1900) format (dd/mm/yyyy)** |
| WEIGHT | **Weight** | **Weight of patient in KG metric** | **Number: range(0,999)** |
| HEIGHT | **Height** | **Height Of Patient in CM metric** | **Number: range (0,300), Not Null** |
| BMI | **Body Mass Index** | **Calculated with formula (Weight)/(Height)^2 to 1 d.p** | **Number: range(> 0)** |
| SYSTOLIC\_BP | **Systolic Blood Pressure** | **Blood pressure when patients heart contracts** | **Number: range(> 0)** |
| DIASTOLIC\_BP | **Diastolic Blood Pressure** | **Blood pressure when patients heart relaxes** | **Number: range( > 0 and < SYSTOLIC\_BP)** |
| AH\_MEDICATION | **Anti-Hyper Sensitive Medication** | **Patient takes/requires hyper-sensitive medication** | **Boolean: Yes / No** |

**ASSESMENTS (assessment\_id, assessment\_date, patient\_id,);**

|  |  |  |  |
| --- | --- | --- | --- |
| ASSESSMENTS | | | |
| ATTRIBUTE NAME | **ATTRIBUTE NAME** | **MEANING** | **DOMAIN DEFINITION** |
| ASSESSMENT\_ID | **Assesment id for patient** | **ID to identify the assessment and can be used to identify doctors responsible ass** | **Unique for each date and to identify doctor and patient involved** |
| ASSESSMENT\_DATE | **Date of assessment** | **Date assessment will take place** | **Date: range from (1st January 1900) format (dd/mm/yyyy)** |
| PATIENT\_ID | **Patient id** | **ID of patient involved as a Foreign Key from patients table.** | **Patient ID for patient in the assessment** |

List the relations with indicated PKs (underlined) and FKs (in italics).

1. Using your Logical Data Model, create a **Physical Data Model**. Prepare a **DDL script to create the table(s) with the appropriate constraints (entity, referential integrity, and domain)**. The BMI column should be defined as a virtual column (value generated automatically based on other columns). Calculate the Body Mass Index (BMI) of the patients at the time of the study. Use only as many significant digits as is justified. **BMI = weight(kg)/height(m)2**

**Print the DDL script.**

**CREATE TABLE PATIENTS (**

**patient\_id VARCHAR2 (10),**

**gender VARCHAR2(1)**

**CONSTRAINT gender\_values CHECK (gender IN ('M','F','U')),**

**dob DATE,**

**weight NUMBER(9,2) NOT NULL**

**CONSTRAINT weight\_check CHECK (weight BETWEEN 0 AND 999),**

**height NUMBER(9,0) NOT NULL**

**CONSTRAINT height\_check CHECK (height BETWEEN 0 AND 300),**

**bmi NUMBER(9,0) AS (weight/(height/100\*height/100)),**

**systolic\_bp NUMBER(9,0)**

**CONSTRAINT systolic\_check CHECK (systolic\_bp>0),**

**diastolic\_bp NUMBER(9,0)**

**CONSTRAINT diastolic\_check CHECK (diastolic\_bp>0),**

**ah\_medication VARCHAR2 (1)**

**CONSTRAINT ah\_taken\_booleab CHECK (ah\_medication IN ('Y','N'))**

**);**

**CREATE TABLE ASSESSMENTS (**

**assessment\_id VARCHAR2 (10) PRIMARY KEY,**

**assessment\_date DATE,**

**patient\_id VARCHAR2 (10),**

**CONSTRAINT patient\_id\_fk FOREIGN KEY (patient\_id) REFERENCES patients (patient\_id)**

**);**

1. Using your table(s) created in 2 perform the following tasks
2. Write SQL DML statements to add **3 patients**. For each patient add data for **3-4 clinical assessments.**

**INSERT INTO PATIENTS (**

**PATIENT\_ID, GENDER, DOB, WEIGHT, HEIGHT, SYSTOLIC\_BP, DIASTOLIC\_BP, AH\_MEDICATION)**

**VALUES (**

**'01',**

**'M',**

**to\_date('01/09/1996','dd/mm/yyyy'),,**

**75,**

**180,**

**50,**

**50,**

**‘Y’);**

**INSERT INTO ASSESSMENTS**

**VALUES (**

**'04\_A',**

**to\_date ('16/06/2017','dd/mm/yyyy'),**

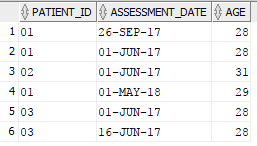
**'03');**

1. Write an SQL query to list the patients and their age at the time of each assessment. List the patient ID, age, and assessment date.

**SELECT DISTINCT patients.patient\_id , assessment\_date,**

**to\_char (assessments.assessment\_date,'yyyy')-to\_char (patients.dob,'yyyy') AS AGE FROM**

**PATIENTS JOIN ASSESSMENTS ON PATIENTS.patient\_id = ASSESSMENTS.patient\_id;**



1. Write an SQL query to list all patients’ BMI characteristics: min, max, median, mean, and standard deviation.

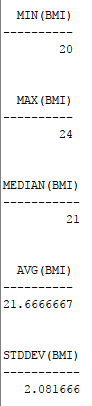
**MIN: SELECT MIN (bmi) FROM patients;**

**MAX: SELECT MAX (bmi) FROM patients;**

**MEDIAN: SELECT MEDIAN (bmi) FROM patients;**

**MEAN: SELECT AVG (bmi) FROM patients;**

**STANDARD DEVIATION: SELECT STDDEV (bmi) FROM patients;**



**Print the SQL statements for a, SQL queries and results for b and c.**

**Part 3** Research (1 %)

**Using online documentation for MySQL 5.7 and Oracle 12c, answer the following questions:**

1. Compare and contrast the methods for creating an **automatic unique identifiers** in each DBMS. Put your answer in a short point/paragraph/table format. List references (at least one source in APA style). Check the library for APA style guides (ask the librarians), for example <http://libguides.tru.ca/APA>
2. Compare and contrast the **storage of date, time (precision), and time zones.** Specify the data types for storing date, time, and time zones in each DBMS. How dates are stored internally in each DBMS? What is the maximum range for dates? Prepare your answer in a table format. List references (at least one source in APA style).

**Submission of the Assignment**

Please create a pdf document with your answers. Make sure that the queries are printed using a **monospaced font (e.g., Courier font)** and are **properly formatted** (see textbook). Create a pdf document and upload the document to the BLearn. Indicate the Part and Problem number for your answers. Your document should have a title page with your name, student number, course title, and date. Upload pdf file.

|  |  |
| --- | --- |
| DBMS | Auto Increment Description |
| ORACLE 12C | Oracle 12C introduces a new method to automatically increment the value for a given column in table. It also allows the user to enforce whether the values are to be unique, if a user can alter them and if they can take NULL values.  GENERATED  [  ALWAYS | BY DEFAULT [ ON NULL ]]  AS IDENTITY [ ( identity\_options ) ]  **GENERATED ALWAYS AS IDENTITY [(IDENTITY\_OPTIONS)]:** This will cause the column to be an identity column and no insert/update statements can alter the value in this row or column.  **GENERATED BY DEFAULT AS IDENTITY [(IDENTITY\_OPTIONS)]:** By default: allows a user to insert a specific value for this column in this field otherwise the value is generated by default. No Null value is allowed however **GENERATED BY DEFAULT ON NULL AS IDENTITY [(IDENTITY\_OPTIONS)]:** ON NULL allows the user to enter a value with a NULL value for this column  http://www.oraclebuffer.com/oracle-internals/oracle-database-12c-new-feature-identity-column-or-auto-increment-column/ |
| ORACLE 11G | Before Oracle 12C, to implement auto increment for a field, a trigger and a sequence had to be created. The trigger does the automation and is fired whenever a row is entered depending on the condition. The sequence keeps count of the row number and  https://chartio.com/resources/tutorials/how-to-define-an-auto-increment-primary-key-in-oracle/ |
| MICROSOFT ACCESS | Microsoft access provides AUTO NUMBER for a data type which causes the field to automatically increment whenever a new row is entered into the table.  https://www.itsupportguides.com/knowledge-base/office-2013/access-2013-how-to-auto-increment-table-primary-key/ |
| SQL SERVER | SQL Server uses IDENTITY key word to cause the field to auto increment whenever a row is entered into the table.  IDENTITY [ (start , increment) ]  https://chartio.com/resources/tutorials/how-to-define-an-auto-increment-primary-key-in-sql-server/ |
| MySQL | Uses AUTOINCREMENT to automatically increment a field whenever a row is entered into the table. |
| POSTGRE SQL | Uses SERIAL Keyword to define a field to be automatically incremented whenever a row is entered into a table.  Also supports sequences in case a custom increment is needed. |

|  |  |
| --- | --- |
| DBMS | Date |
| Oracle | * Date and time both stored in DATE type * You can specify a fractional precision of up to nine digits for any TIMESTAMP type * TIMESTAMP can store time zone but needs to be explicitly listed * The date and time values is between 4712 BC through to 9999 AD. |
| MySQL | * DATE stores dates from 1-Jan-1000 through 31-Dec-9999. * TIME stores hour/minute/second values from −838:59:59 through 838:59:59 * DATETIME stores both date and time of day ranging same as DATE and TIME except the hours are 23hrs long. * TIMESTAMP stores UNIX timestamp values and is set to the time an INSERT or UPDATE statement is processed unless you explicitly define a custom value. |
| PostGre SQL | * DATE stores a date only. * TIME types store time of day. * TIMESTAMP types store both date and time. * The range of years is from 4713 B.C. through 294,276 A.D. (TIMESTAMPs) and 5,874,897 A.D. (DATEs). * TIME and TIMESTAMP allow you to limit the number of precision digits for fractional seconds in DOUBLE PRECISION or BIGINT. * DOUBLE PRECISION floating point (0 to 6) or BIGINT (0 to 10). |
| SQL Server | * DATE stores date only from 1-Jan-0001 through 31-Dec-9999. * DATETIME stores date and time of day to an increment of 3.33 milliseconds, with a range of 1-Jan-1753 through 31- Dec-9999. * DATETIME2 is a combination of DATE and TIME. * DATETIMEOFFSET extends DATETIME2 with a time zone offset. * SMALLDATETIME stores date and time of day to the minute, with a range of 1-Jan-1900 through 6-Jun-2079. TIME stores time of day. * DATETIME2, DATETIMEOFFSET, and TIME take an optional parameter to specify the decimal precision of the second’s value. The default precision is to store seconds to seven decimal places. The valid range is from 0 through 7. |
| MS Access | * Date/Time stored in a single data type DATE/TIME * Date and Time values for the years 100 through 9999. * Valid date values range from -657,434 (January 1, 100 A.D.) to 2,958,465 (December 31, 9999 A.D.). Valid time values range from .0 to .9999, or 23:59:59. |

Date and time data got from SQL pocket guide by Jonathan Gennick