Part 2 – INSERTING some data of your own (5%)

1.

INSERT a new Customer with your first name for the value of “fname”, your last name for the

value of “lname”, and your PSID for the value of “Username”. You can put any reasonable value

you want for the value of phone number.

INSERT INTO customers (username, fname, lname, phone) VALUES ('2093536', 'Shashank', 'Bhushan', '(832) 933-8898');

2.

Update the Horses table so that the new customer you created in the previous step (you) is the

owner for the horse name “Flash”.

UPDATE

horses SET owner='2093536'

WHERE

name='Flash';

3.

Update the CourseCustomer table so that the new customer you created (you) is taking the

following courses:

a.

Introduction to Racing in Summer of 2024

-- Find the course ID for Introduction to Racing in Summer of 2024

SELECT COURSEID

FROM Courses

WHERE FKTOPICCODE = (SELECT TOPICCODE FROM Topics WHERE

COURSENAME = 'Intro to Racing')

AND TERM = 'Summer'

AND YEAR = 2024;

-- Insert for Intro to Racing

INSERT INTO CourseCustomer (FKUSERNAME, FKCOURSEID)

SELECT '2093536', COURSEID

FROM Courses

WHERE FKTOPICCODE = (SELECT TOPICCODE FROM Topics WHERE

COURSENAME = 'Intro to Racing')

AND TERM = 'Summer' AND YEAR = 2024;

b.

Introduction to Polo in Fall of 2024

-- Find the course ID for Introduction to Polo in Fall of 2024

SELECT COURSEID

FROM Courses

WHERE FKTOPICCODE = (SELECT TOPICCODE FROM Topics WHERE

COURSENAME = 'Introduction to Polo')

AND TERM = 'Fall'

AND YEAR = 2024;

-- Insert for Introduction to Polo

INSERT INTO CourseCustomer (FKUSERNAME, FKCOURSEID)

SELECT '2093536', COURSEID

FROM Courses

WHERE FKTOPICCODE = (SELECT TOPICCODE FROM Topics WHERE

COURSENAME = 'Introduction to Polo')

AND TERM = 'Fall' AND YEAR = 2024;

(Note: you will need to find the course names in the “Topics” table, then

find the CourseID for the particular term and year in the “Courses” table)

Part 3 – Basic Single Table SQL (10%)

1. Return Name, Title, and Salary for workers than have a salary over 100,000. Order is not important.

**SELECT** Name, Title, Salary

**FROM** WORKERS *w*

**WHERE** Salary > 100000;

2. Return all attributes for horses that have a weight greater than or equal to 1800 and are Female.

**SELECT** \*

**FROM** Horses

**WHERE** weight >= 1800 **AND** sex = **'F'**;

3. Return the name and title of workers whose title starts with the word “Director”

**SELECT** Name, Title

**FROM** workers

**WHERE** Title **LIKE** **'Director%'**;

4. Return all attributes for horses that have “r” as the second letter of their name

**SELECT** \*

**FROM** Horses

**WHERE** **SUBSTR**(Name, 2, 1) = **'r'**;

5. Return the name and title for all workers that are no longer working for MGHH (i.e., they have a

value for the attribute TerminationDate).

**SELECT** Name, Title

**FROM** Workers

**WHERE** TerminationDate **IS** **NOT** **NULL**;

Part 4 - SQL with Aggregate Functions (30%)

1. Return the number of workers for each job title ordered alphabetically by job title.

**SELECT** Title, **COUNT**(\*) **AS** *"Number of Emps"*

**FROM** Workers

**GROUP** **BY** Title

**ORDER** **BY** Title;

2. Find the average weight of horses for each value of Sex. Round the result to one decimal place.

**SELECT** Sex, **ROUND**(**AVG**(Weight), 1) **AS** *"Average Weight"*

**FROM** Horses

**GROUP** **BY** Sex;

3. Find the number of horses of each color. Return the results so that the color with the most horses is

first, and the color with the fewest horses is last.

**SELECT** Color, **COUNT**(\*)

**FROM** Horses

**GROUP** **BY** Color

**ORDER** **BY** **COUNT**(\*) **DESC**;

4. Return the number of horses for each combination of color and sex. Return the results

alphabetically by color, then sex within color.

**SELECT** Color, Sex, **COUNT**(\*)

**FROM** Horses

**GROUP** **BY** Color, Sex

**ORDER** **BY** Color, Sex **DESC**;

5. Find the number of horses of each color for horses that are female and do not have spots. Return

the results so that the color with the most horses is first, and the color with the fewest horses is last.

**SELECT** Color, **COUNT**(\*)

**FROM** Horses

**WHERE** Sex = **'F'** **AND** Spots = **'No'**

**GROUP** **BY** Color

**ORDER** **BY** **COUNT**(\*) **DESC**;

6. Return the number of workers for each job title ordered alphabetically by job title. Do not include

workers than have been fired (i.e., those that have a value for the attribute “TerminationDate”)

**SELECT** Title, **COUNT**(\*) **AS** *"Number of Emps"*

**FROM** Workers

**WHERE** TerminationDate **IS** **NULL**

**GROUP** **BY** Title

**ORDER** **BY** Title;

7. Return the number of stables for each barn that currently have a status of “Maintenance”

**SELECT** FKBARNID, **COUNT**(\*) **AS** *"MaintCount"*

**FROM** Stables

**WHERE** STATUS = **'Maintenance'**

**GROUP** **BY** FKBARNID;

8. Return last names that are shared by two or more customers. Order the results so the most popular

last name is first.

**SELECT** LNAME , **COUNT**(\*)

**FROM** Customers

**GROUP** **BY** LNAME

**HAVING** **COUNT**(USERNAME) > 1

**ORDER** **BY** **COUNT**(\*) **DESC**;

9. List each job title and the average salary, rounded to zero decimal places, minimum salary,

maximum salary, and number of employees for each job title. Order the results with the highest salary

at the top.

**SELECT** TITLE,

**ROUND**(**AVG**(Salary), 0) **AS** *AVGSalary*,

**MIN**(Salary) **AS** *MinSalary*,

**MAX**(Salary) **AS** *MaxSalary*,

**COUNT**(\*) **AS** *Employees*

**FROM** Workers

**GROUP** **BY** TITLE

**ORDER** **BY** **AVG**(Salary) **DESC**;

10. Return the total number of horses in the database, as well as the number of horses that are

owned by someone. You may only use the data in the “horses” table (do not do a join to the

customers table). Check the answer with professor-Cleared as 1 more then total count in screenshot.

**SELECT**

**COUNT**(\*) **AS** *"Total Horses"*,

**SUM**(**CASE** **WHEN** OWNER **IS** **NOT** **NULL** **THEN** 1 **ELSE** 0 **END**) **AS** *"Owned Horses"*

**FROM** Horses;

Part 5 – Binary Queries (21%)

1. Return the name of the horse and first name, last name, and phone number of the owner for all Female horses whose owner’s phone number is in the (615) area code. Order by the horse’s name.

**SELECT** *h*.Name **AS** *Name*,

*c*.FNAME **AS** *FNAME*,

*c*.LNAME **AS** *LNAME*,

*c*.PHONE **AS** *PHONE*

**FROM** Horses *h*

**INNER** **JOIN** Customers *c* **ON** *h*.OWNER = *c*.USERNAME

**WHERE** *h*.Sex = **'F'** **AND** *c*.PHONE **LIKE** **'(615)%'**

**ORDER** **BY** *h*.Name;

2. Return the course name and name of the instructor for all courses offered in Summer of 2024

**SELECT** *t*.COURSENAME, *w*.NAME, *c*.TERM, c.**YEAR**

**FROM** Courses *c*

**INNER** **JOIN** Topics *t* **ON** *c*.FKTOPICCODE = *t*.TOPICCODE

**INNER** **JOIN** Workers *w* **ON** *c*.FKEMPID = *w*.EMPID

**WHERE** *c*.TERM = **'Summer'** **AND** c.**YEAR** = 2024

**ORDER** **BY** *t*.COURSENAME **DESC**,*w*.NAME **ASC**;

3. Return the number of horses owned by each customer than owns more than 4 horses. Order by county of horses in descending order.

**SELECT** cus.USERNAME, cus.FNAME, cus.LNAME,**COUNT**(\*)

**FROM** Horses h

**INNER** **JOIN** Customers cus **ON** h.OWNER = cus.USERNAME

**GROUP** **BY** cus.USERNAME , cus.FNAME, cus.LNAME

**HAVING** **COUNT**(\*) > 4

**ORDER** **BY** **COUNT**(\*) **DESC**;

4. For each certificate, list all the workers that hold that certificate. List all certificates even if no workers have them (Note: there are 54 rows returned, only the first 10 are shown here)

**SELECT** *c*.CERTNAME, *w*.NAME, *w*.TITLE

**FROM** Certifications *c*

**LEFT** **JOIN** Workercert *wc* **ON** *c*.CERTID = *wc*.FKCERTID

**LEFT** **JOIN** Workers *w* **ON** *wc*.FKEMPID = *w*.EMPID

**GROUP** **BY** *c*.CERTNAME, *w*.NAME, *w*.TITLE

**ORDER** **BY** *c*.CERTNAME, *w*.TITLE;

5. Using a recursive join, generate this org chart. Only include employees that are currently working for MGHH.

**SELECT** *e2*.NAME **AS** *"Manager Name"*, *e1*.name **AS** *"Employee"*

**FROM** workers *e1*, workers *e2*

**WHERE** *e1*.supervisor = *e2*.EMPID **AND** *e1*.TerminationDate **IS** **NULL**;

6. Return horse name, medication name, dose, and frequency for all horses that have been ordered to take a “Dewormer” medication. Order alphabetically by medication name, then horse name.

**SELECT** *h*.NAME,

*m*.MEDNAME,

*o*.DOSE,

*o*.FREQUENCY

**FROM** Horses *h*

**INNER** **JOIN** Orders *o* **ON** *h*.NAME = *o*.ORD\_HORSENAME

**INNER** **JOIN** Medications *m* **ON** *o*.ORD\_MEDCODE = *m*.MEDCODE

**WHERE** *m*.CLASSIFICATION = **'Dewormer'**

**ORDER** **BY** *m*.MEDNAME, *h*.NAME;

7. Return the name of all workers that teach a class and are also assigned to oversee one or more barns.

**SELECT** *w*.NAME

**FROM** Workers *w*

**INNER** **JOIN** Courses *c* **ON** *w*.EMPID = *c*.FKEMPID

**INNER** **JOIN** WorkerBarn *wb* **ON** *w*.EMPID = *wb*.FKEMPID

**GROUP** **BY** *w*.NAME

**HAVING** **COUNT**(**DISTINCT** *wb*.FKBARNID) >= 1;

Part 6 – Using Multiple Approaches (24%)

1. Return the Name, spots, color, and username of the owner for horses that are the color Red and have spots using an “AND” operator in your WHERE statement.

**SELECT** *h*.NAME, *h*.SPOTS, *h*.COLOR, *h*.OWNER

**FROM** Horses *h*

**WHERE** *h*.COLOR = **'Red'**

**AND** *h*.SPOTS = **'Yes'**;

2. Return the Name, spots, color, and username of the owner for horses that are the color Red and have spots using a set operation.

**SELECT** **DISTINCT** *ho*.NAME, *ho*.SPOTS, *ho*.COLOR, *ho*.OWNER

**FROM** Horses *ho*

**JOIN** CUSTOMERS *c* **ON** *c*.USERNAME = *ho*.OWNER

**WHERE** *ho*.COLOR = **'Red'**

**AND** *ho*.SPOTS = **'Yes'**

**INTERSECT**

**SELECT** *ho*.NAME, *ho*.SPOTS, *ho*.COLOR, *ho*.OWNER

**FROM** Horses *ho*

**JOIN** CUSTOMERS *c* **ON** *c*.USERNAME = *ho*.OWNER;

3. Return the Name, spots, color, and username of the owner for horses that are the color Red and have spots using a subquery.

**SELECT** *h*.name, *h*.spots, *h*.color, *h*.owner

**FROM** Horses *h*

**WHERE** *h*.color = **'Red'**

**AND** *h*.spots = **'Yes'**

**AND** *h*.owner **IN** (

**SELECT** owner

**FROM** Horses

**WHERE** color = **'Red'**

**AND** spots = **'Yes'**

);

4. Find all workers who are teaching a class but do not have the teaching certification. Use a subquery.

**SELECT** *w*.NAME

**FROM** Workers *w*

**WHERE** *w*.EMPID **IN** (

**SELECT** FKEMPID

**FROM** Courses *c*

)

**AND** *w*.EMPID **NOT** **IN** (

**SELECT** FKEMPID

**FROM** WorkerCert *wc*

**WHERE** *wc*.FKCERTID = **'TEA'**

)

**ORDER** **BY** *w*.NAME;

5. Find all workers who are teaching a class but do not have the teaching certification. Use the MINUS set operation.

**SELECT** Workers.NAME

**FROM** Workers

**INNER** **JOIN** Courses *c* **ON** Workers.EMPID = *c*.FKEMPID

**MINUS**

**SELECT** Workers.NAME

**FROM** Workers

**INNER** **JOIN** Courses c **ON** Workers.EMPID = c.FKEMPID

**INNER** **JOIN** Workercert wc **ON** Workers.EMPID = wc.FKEMPID

**WHERE** wc.FKCERTID **LIKE** **'TEA%'**;

6. Return all customers than have not enrolled in a course using a subquery

**SELECT** *c*.USERNAME, *c*.FNAME, *c*.LNAME, *c*.PHONE

**FROM** Customers *c*

**WHERE** *c*.USERNAME **NOT** **IN** (

**SELECT** FKUSERNAME

**FROM** Coursecustomer

);

7. Return all customers than have not enrolled in a course using an outer join

<<Same result as question above>>

**SELECT** *c*.USERNAME, *c*.FNAME, *c*.LNAME, *c*.PHONE

**FROM** Customers *c*

**LEFT** **JOIN** Coursecustomer *cc* **ON** *c*.USERNAME = *cc*.FKUSERNAME

**WHERE** *cc*.FKUSERNAME **IS** **NULL**;

8. Return the username of all customers than have not enrolled in a course using a set operation (Check with Salman)

**SELECT** ***c***.USERNAME

**FROM** Customers c

**MINUS**

**SELECT** FKUSERNAME

**FROM** COURSECUSTOMER;

Part 7 – String Manipulation (15%)

1. Return each worker’s name as it is stored in the database, and the name formatted as “Lastname,Firstname” for workers that have a title “Ranch Hand”

**SELECT** NAME

, **CONCAT**(**SUBSTR**(NAME, **INSTR**(NAME, **' '**) + 1),**CONCAT**( **', '**, **SUBSTR**(NAME, 1, **INSTR**(NAME, **' '**) - 1))) **AS** *"Formatted Name"*

**FROM** Workers

**WHERE** TITLE = **'Ranch Hand'**;

2. Return the number of customers for each area code (note: the area code is the first part of the phone number found in parentheses). Return the results with the most popular area code first.

**SELECT** **SUBSTR**(PHONE, 2, **INSTR**(PHONE, **')'**) - 2) **AS** *AreaCode*, **COUNT**(\*) **AS** *"COUNT(\*)"*

**FROM** Customers

**HAVING** **COUNT**(\*) >1

**GROUP** **BY** **SUBSTR**(PHONE, 2, **INSTR**(PHONE, **')'**) - 2)

**ORDER** **BY** **"COUNT(\*)"** **DESC**;

3. Return the names of all worker along with their phone number formatted as (xxx) xxx-xxxx.,

**SELECT** NAME

, **CONCAT**(**'('**,**CONCAT**(**LPAD**(**SUBSTR**(PHONE, 1, 3), 3, **'0'**),**')'**)) || **' '** || **LPAD**(**SUBSTR**(PHONE, 4, 3), 3, **'0'**) || **'-'** || **SUBSTR**(PHONE, 7, 4) **AS** *FormattedPhone*

**FROM** Workers;

4. Return the number of races held in each state

**SELECT** **SUBSTR**(LOCATION, -2, 2) **AS** *State*, **COUNT**(\*) **AS** *Numraces*

**FROM** Races

**GROUP** **BY** **SUBSTR**(LOCATION, -2, 2)

**ORDER** **BY** *Numraces*, *State*;

5. Return the first and last name and a “masked” version of their phone number than shows only the last four digits, like this: (\*\*\*) \*\*\*-1234 for customers with a last name that starts with A or R. (Ask for extra \* in masked number from professor)

**SELECT** FNAME, LNAME,

**CONCAT**(**'(\*\*\*) \*\*\*-'**,**SUBSTR**(PHONE, -4)) **AS** *Mask*

**FROM** Customers

**WHERE** **UPPER**(**SUBSTR**(LNAME, 1, 1)) **IN** (**'A'**, **'R'**);

Part 8 – Bonus (12% extra)

1. Find the average number of workers assigned to each type of barn. Order by the average number of workers in descending order. Round to two decimal places.

**SELECT** *b*.DESCRIPTION,

**ROUND**(**COUNT**(\*)/**COUNT**(**DISTINCT** *wb*.FKBARNID), 2) **AS** *"Average Workers"*

**FROM** WorkerBarn *wb*

**INNER** **JOIN** Barns *b* **ON** *wb*.FKBARNID = *b*.BARNID

**GROUP** **BY** *b*.DESCRIPTION

**ORDER** **BY** *"Average Workers"* **DESC**;

2. Using a hierarchical query, generate this org chart (complete with indentions) using the data in the workers table. Only include employees that are currently working for MGHH.

**SELECT** **RPAD**(**' '**, (**LEVEL**-1)\*2, **' '**) || Name **AS** *"Org"*, Title

**FROM** Workers

**WHERE** TerminationDate **IS** **NULL**

**CONNECT** **BY** **PRIOR** EmpID = Supervisor

**START** **WITH** Supervisor **IS** **NULL**

**ORDER** SIBLINGS **BY** Title;

3. Return each employee’s empid, name, hire date, and number of years employed for workers that are still working for MGHH. Be sure to format the hired date appropriately. Note that your results may vary SLIGHTLY depending on when you run the query.

**SELECT** empid, name, **TO\_CHAR**(hiredate,**'MM/DD/YYYY'**) **AS** *"Hired On"*,

**ROUND**(**TO\_CHAR**((**SYSDATE** - hiredate) / 365), 1) **AS** *"Years Employed"*

**FROM** Workers

**WHERE** terminationdate **IS** **NULL**

**ORDER** **BY** hiredate **ASC**;

4. Generate this report of courses being offered, the instructor and their job title, the current enrollment, and the calculated percentage full.

**SELECT**

*t*.CourseName,

*w*.Name || **' ('** || *w*.Title || **')'** **AS** *Instructor*,

*c*.Term || **' '** || c.**Year** **AS** *Term*,

*c*.SizeLimit,

**COUNT**(\*) **AS** *Enrollment*,

**ROUND**(**COUNT**(\*) / *c*.SizeLimit \* 100) || **'%'** **AS** *PercentFull*

**FROM**

Courses *c*

**INNER** **JOIN**

Topics *t* **ON** *c*.FKTopicCode = *t*.TopicCode

**INNER** **JOIN**

Workers *w* **ON** *c*.FKEmpID = *w*.EmpID

**LEFT** **JOIN** -- Changed INNER JOIN to LEFT JOIN

CourseCustomer *cc* **ON** *cc*.FKCourseID = *c*.CourseID

**GROUP** **BY**

***c***.***CourseID***, ***t***.***CourseName***, ***w***.***Name***, ***w***.***Title***, ***c***.***Term***, c.**Year**, *c*.SizeLimit

**ORDER** **BY**

*c*.Term || **' '** || c.**Year** **ASC**,

t.CourseName, **COUNT**(\*) **DESC**;

5. Return the name of the horse, the owner’s first and last name (concatenated, with an attribute name of “Owner”), the year, place, and racename for all horses that placed 1, 2, or 3 in any race in 2001. Order alphabetically by racename, and numerically by place within racename. Be sure to include horses even if they do not have an owner.

**SELECT** *h*.Name,

(**CASE** **WHEN** *c*.username **IS** **NULL** **THEN** **' '** **ELSE** *c*.Fname || **' '** || *c*.Lname **END**) **AS** *"Owner"*,

hr.**year**,

*hr*.place,

*r*.RaceName

**FROM** horses *h*

**LEFT** **JOIN** customers *c* **ON** *h*.owner = *c*.username

**JOIN** HorseRaces *hr* **ON** *h*.Name = *hr*.FKHorseName

**JOIN** races *r* **ON** *hr*.FKRaceID = *r*.RaceID

**WHERE** hr.**year** = 2021 **AND** *hr*.place **IN** (**'1'**, **'2'**, **'3'**)

**ORDER** **BY** *r*.RaceName, *hr*.place;

6. Return the total number of top-10 finishes each customer has had across all the horses they own

**SELECT** **RPAD**(**' '**, (**LEVEL**-1)\*2, **' '**) || Name **AS** *"Org"*, Title

**FROM** Workers

**WHERE** TerminationDate **IS** **NULL**

**CONNECT** **BY** **PRIOR** EmpID = Supervisor

**START** **WITH** Supervisor **IS** **NULL**

**ORDER** SIBLINGS **BY** Title;