INFO 6210
Data Management and Database Design
Practice Exam Questions

Professor: Nik Bear Brown

Exam is Tuesday April 9, 2019
One page cheat sheet. No books, computer or phone.

Assume the following four tables (users, tags, tweets, and twitter_jobs) exist with the following fields
PK means primary key.

```
-- TABLE users
-- user_id_str (PK)
-- user_name,user_description,user_favourites_count,user_followers_count,
-- user_friends_count,user_location,user_profile_image_url,user_screen_name,
-- user_statuses_count,user_time_zone
--

-- TABLE tags
-- id_str
-- hashtags, urls

-- TABLE tweets
-- id_str (PK)
lang,created_at,tweet,retweeted,user_id_str,user_timezone_country,polarity,sentiment

-- TABLE twitter_jobs
-- id (PK)
id_str, logged ( a timestamp(6))
-- ****************************************************************
```

Write SQL statements to do the following on the above tables:

1. Select columns

```
-- Answer Question 1
-- SELECT count(*) from twitter
-- SELECT * FROM twitter limit 10
```

2. Filter rows

```
-- Answer Question 2
-- select count (*) from twitter
```

```
-- where lang like 'en'
```

3. Sort your query

```
-- Answer Question 3
-- SELECT user_name , COUNT(user_id_str)
-- from users
-- GROUP BY 1
-- ORDER BY 2 DESC
```

4. Group by an attribute

```
-- Answer Question 4
--SELECT count(*), date_trunc('hour', created_at)
--FROM tweets
--GROUP BY 2


-- Answer Question 4
-- SELECT user_name, date_trunc('hour', created_at)
-- FROM twitter
-- GROUP BY 2
-- ORDER BY 2
```

5. Calculate an aggregate function on an attribute of the group.

```
-- Answer Question 5
-- SELECT user_screen_name,user_name, MAX(user_followers_count)
-- FROM twitter
-- WHERE lang like 'en'
-- GROUP BY 1,2
-- ORDER BY 3 DESC
```

6. Use DISTINCT keyword will make it so it only returns one instance of each attribute.

```
-- Answer Question 6
--SELECT DISTINCT user_name from users
--WHERE user_time_zone LIKE 'Eastern Time (US & Canada)'
```

7. Create a column that is calculated from other columns.

```
--SELECT user_favourites_count,user_followers_count, (user_favourites_count + user_followers_count)
AS total_count from users
```

OR

```
-- Answer Question 7, 12
-- CREATE FUNCTION verified(twitter) RETURNS TEXT AS
```

```
-- $$ SELECT CASE WHEN $1.user_followers_count > 200000 THEN 'Yes'
--
--         ELSE 'NO'
--
-- END;
-- $$ LANGUAGE SQL IMMUTABLE;
-- SELECT user_name, user_followers_count, t.verified FROM twitter t;
```

8. Count all of the null values in a nullable field.

```
-- Answer Question 8
-- SELECT COUNT(*) from twitter
-- WHERE user_location IS NULL;
```

9. In a text field count all of rows in the columns that contain the letter 'a'.

```
-- Answer Question 9
-- SELECT COUNT(*) FROM twitter
-- WHERE user_name LIKE '%a%'
```

10. Subselect columns using a subquery.

```
-- Answer Question 10
-- SELECT user_name, user_screen_name, user_followers_count
-- FROM twitter
-- WHERE user_followers_count >
-- (SELECT AVG(user_followers_count) from twitter);
```

11. Computationally what is the most expensive operation in the relational data model?

```
-- Answer Question 11
-- Joins are considered the most computationally heavy operations, but without joins there wont be any
need to have a relational database.
```

12. Write a function to calculate something.

```
DELIMITER //


CREATE FUNCTION getRetweet(idm  varchar(20))

RETURNS FLOAT

BEGIN
```

```
    DECLARE pm FLOAT;

    SET pm:=0;

    SELECT tweets into pm from tweets where id=idm;

    RETURN pm;

    END//
```

```
-- Answer Question 12 (and 20)
-- CREATE FUNCTION insert_table(_location text,_title text,_company text,_salary double
precision,_summary text,_salary_duration text,_count integer)
-- RETURNS void AS
-- $BODY$
-- BEGIN
--         INSERT INTO indeed_final(location, title, company, salary, summary, salary_duration, count)
--         VALUES (_location,_title, _company, _salary, _summary, _salary_duration, _count);
-- END;
-- $BODY$
-- LANGUAGE 'plpgsql' VOLATILE
-- COST 100;
-- SELECT * FROM insert_table('Atlanta, GA', 'Data Analyst Intern',  'Cox Communications', NULL, 'Applies
quantitative methods of social science data analysis. May write code to automate reports and templates
and consolidate data into reports and knowledge....',NULL,NULL);
-- SELECT * FROM indeed_final
-- WHERE company LIKE 'Cox Communications';
```

13. When two tables are joined in a relational database what is the resulting data structure?

```
-- Answer Question 13
-- When two tables are joined the resulting data structure is another table.
-- When we join two tables, we get a new table with all the columns that we selected in the query
```

14. Select and filter some data from a table created by a join.

```
-- Answer Question 14
-- SELECT users.user_name , COUNT(users.user_name)
-- FROM users
-- INNER JOIN twitter ON twitter.user_id_str = users.user_id_str
-- WHERE twitter.lang LIKE '%en%'
-- GROUP BY 1
-- ORDER BY 2 DESC
```

15. Why not put all the data in one big table and avoid all of these joins?

-- Answer Question 15

Relationship

1. If the relationship between multiple colums is one-to-one then it better to store the data in one table, since it reduces the number of joins the table has to do.
2. If the relationship between two tables in one-to-many, then it will be better to split into seperate tables to reduce duplicate data. Duplicate data wastes lots of storage and cache space and makes database harder to maintain.

Computation

1. We should always start by 3NF form and only denormalise if we find a specific performance problem.
2. Also, by storing all the data in a single table, makes querying the table expensive as querying a large table for a single value is more costly than querying two small tables.
3. Also, by storing it in one table and duplicating data, we run a risk of allowing inconsistant data to be inserted into database thus nullifying one of the core properties of Relational databases (ACID)
4. When we store all the data in a single large table, if a remote server needs a single value for the table, entire table needs to be transferred through network, which increases network transmission times. If the data is stored in normalised form, this can be avoided.

Thus, even though Joins are expensive, there are many factors to be considered while designing a database. Joins are needed only when we need data from two tables, where rest of the above reasons together combined can be more expensive than joins. Also, we can reduce repeated joins by creating a view of the table with joins and query it.

16. Why create views?

-- Answer Question 16

Creating views has several benefits.

1. Views can hide complexity

If we have a query that requires joining several tables or has a complex logic/calculations, we can create a view and query it just like tables. Thus, a view is encapsulation of a complex or expensive query.

2. Views can be used as a security mechanism

With a view, we can select certain columns and/or rows from a table, and can set permissions on the view instead of the underlying tables. This allows surfacing only the data that a user needs to see and rest of the data can be hidden from the user.

3. To Denormalize data for reporting purpose

We can use a view to denormalize and/ or aggregate the tables. This results in reducing the redundancy in writing the queries also maximizing the performance of the database.

4. Refactoring Database

We can hide the change so that the old code does not see it by creating a view.

17. Select and filter some data from a table created by a view.

```
-- Answer Question 17
-- CREATE VIEW jobs_at_amazon AS
-- SELECT * FROM indeed_final
-- WHERE company LIKE '%Amazon%';
-- SELECT * FROM jobs_at_amazon;
```

18. Why create temporary tables?

```
-- Answer Question 18
-- Temporary tables can be helpful in cases where you want to perform manipulations on your table but you have only read access.
-- You can pull up a temporary table and perform the necessary manipulations. Also there can be situations in which you would want data to persist
-- only for that session , in such cases temporary tables can be useful.
```

A Temporary Table is only visible to the current session and is dropped automatically once the session ends. This means that two different sessions can use the same temporary table name without conflicting with each other or with an existing non-temporary table of the same name.

This has several advantages,

1. We can pull data from various tables, do some work on that data and then combine everything into one result set.
2. We can use temporary table for tilting the data i.e. turning rows to columns, etc. which we need for advance processing.

19. Select and filter some data from a table created by a temporary table.

```
-- Answer Question 19
-- CREATE TABLE tweets_india AS
-- SELECT * FROM twitter
-- WHERE user_timezone_country LIKE '%India%';

-- SELECT * FROM tweets_india
-- WHERE polarity > 0.5
```

-- AND user_followers_count > 1000;

20. Insert some data in to a table.

Insert into user values('@Bear','Demo Add User', 100000,10 , 9999)"

21. Update some data in a table.

--Question 21
-- UPDATE tags_final
-- SET tags = 'cloudguru'
-- WHERE id = 20;

22. Delete some data a table.

-- Answer Question 22
-- DELETE from tags_final
-- WHERE id = 20;

23. Create a stored procedures or function that does something useful

-- Answer Question 24

```
CREATE FUNCTION diffrt(idm  varchar(20),idb varchar(20))

RETURNS FLOAT

BEGIN

DECLARE pm FLOAT;

DECLARE pmb FLOAT;

SET pm:=0;

SET pmb:=0;

SELECT tweets into  pm from tweets where id=idm;

SELECT tweets into  pmb from tweets where id=idb;

Return pm-pmb;

END//
```

-- CREATE OR REPLACE FUNCTION add_tags(tag text)

```
--    RETURNS void AS $$
--    BEGIN
--    INSERT INTO tags_final (id, tags)
--    VALUES (DEFAULT, tag);
--    END;
--    $$ LANGUAGE plpgsql;
-- SELECT add_tags('paperspace');
-- SELECT * FROM tags_final;
```

24. Create a index on a non-key attribute.

```
--Question 24
--CREATE INDEX faster_parse ON twitter;
```

25. Create a trigger.

```
-- Answer Question 25
-- CREATE TRIGGER log_jobs
-- AFTER INSERT
-- ON twitter
-- FOR EACH ROW
-- EXECUTE PROCEDURE log_jobs();
```

26. Create a transaction.

```
-- Answer Question 26
-- BEGIN ;
-- DELETE from twitter_jobs
-- WHERE urls IS NULL;
-- ROLLBACK;  -- If we replace with commit changes will be made to table
```

27 Explain to an eight year old (i.e. your professor) what are the first three Normal Forms.

```
-- Answer Question 27
-- Coming back to the main point normalization is the process of reducing coupling between columns(ie dependency between each other).
-- Lets say you grow older every year, you wouldnt want to update all your details on all your accounts. If its not normalized then there can be places
-- where your age would never be updated, without you explicitly doing it.
-- 1NF
-- A table in the first normal form should be atomic and have non repeating rows or columns.

--2NF
-- There shouldnt be any partial dependency , which means that no value in the table should be dependant on a part of the primary key.
-- Lets say there is a composite key in a table , but a column in that table is dependant on a part of the composite key. This is in violation of 2NF.
-- Also a table should be 1NF to be classified as 2NF.
```

--3NF
-- The table should be in 2NF to be classified as 3NF.
-- No non primary attribute in the table should be dependant on other non primary attribute in the table. If this is happening,
-- seperate either one and create a key in the other table which will persist in the table.
-- Also known as Transitive Dependency or relying on an unreliable resource(eg. Wikipedia or Wrong Normalization Videos on Youtube)

There are 3 common forms of Normalization. i.e. 1st, 2nd and 3rd Normal Form.

These Normalization rules are progressive i.e. to be in 3rd normal form, the data must satisfy 2nd normal. Similarly, to be in 2nd normal form, the data must satisfy 1st normal form.

First Normal Form

The data is stored in a relational table and each column contains a atomic value i.e only a single value and there are no repeating group of columns i.e collection of columns containing similar kinds of values in a table.

For example, in the case of a table called Person with columns named Name, Family Name, Phone 1, Phone 2, Phone 3, Phone 4 ... etc the collection of columns Phone N is referred to as a repeating group and assumed to be in violation of 1NF even though it is not a repeating group.

Second Normal Form

The data stored in the table is in 1st normal form and all the columns in the table depend on the primary key of the table.

A Primary Key is a field in the table which uniquely identifies every row in the table. All the primary keys must contain a unique record and it cannot be null i.e without a value.

Third Normal Form

The data stored in the table must be in 2nd normal form and all of its columns are non-transitively dependent on the primary key of the table.

By non-transitive dependency, we mean that there should not be any indirect relation between the columns and the primary key of the table.

for example, in a table of College courses, if we have columns like

```
Course_id-----CRN-----Professor_name-----Course_name
```

Here, CRN is dependent on Course_id but Professor_name is dependent on CRN, thus there is a transitive dependency between Professor_name and Course_id i.e. there is an indirect dependency between Professor_name and Course_id. This is a voilation of 3rd Normal Form, thus, it is not acceptable.