University Of Portsmouth
BSc (Hons) Computer Science
First Year

Database Systems Development

M30232
September 2022 - May 2023
20 Credits

Thomas Boxall
up2108121@myport.ac.uk

Contents

S.1. Introduction to module

29-09-22 13:00 Mark RB LT1

The Module Coordinator for this module is Mark (based in BK 3.09), he's assisted by Valentin and Roy in some sessions alongside others too.
Mark is using a new piece of software to make his presentations with, this is currently in the test phases and he may change back to PowerPoint if people don't like it. Slides are available on Moodle as HTML format, they can be printed to PDF files for offline viewing.

Module Aims

This module aims to help you understand where the database sits in modern systems. It does not train us to be database administrators. It gives us the skills to design a database and the knowledge of how to access it and do so safely.
This module will start from the ground up.

Learning Outcomes

- Demonstrate the fundamental principles of database design & development
- Use appropriate analysis techniques to identify the requirements of a database.
- Design and build a relational database, given a set of requirements.
- Understand how to apply data manipulation using SQL.

Historically, this module used to focus on the elements of Computer Science which relate to databases (for example, software development lifecycles). Now, it focuses on just databases.

Content Overview

This module provides an understanding of the theory of relational database design using tools standard to the industry. We will be taught how to design databases using Crows Foot Entity Relationship Diagrams and SQL to create the database. This module will also cover normalisation.

Teaching Overview

The module is a year long, worth 20 credits and has two different styles of teaching.
There will be one, one hour lecture per week. In this session, we will be taught the knowledge which we can put into practice in the following weeks practical session.
There will be one, one and a half hour practical session per week. In this session, we will practice the skills required for databases. *(N.B. This session is timetabled for two hours on the timetable, generally the lecturers will leave after an hour and a half however students can remain in the room until the end of the two hours.)*
If you are unable to make it to a lecture, you need to read the content provided on Moodle. If you are unable to make it to a practical, you need to read and do (most importantly, do) the content on Moodle; this is so you are able to complete the following practical as they all build on each other.

Resources

There are a number of resources talked through:

- Moodle - the universities Virtual Learning Environment. Notes from lectures and from practicals will be uploaded here along with quizzes and other resources.
- Google Virtual Machine - the virtual machine in which our database lives. You do not need the university VPN to access it, as it requires a SSH connection. The data is hosted by Google, the module staff have some control over the machines. More detail on this will be provided in the first practical session.
- Google Workspace
- Microsoft Office. This is available free from the university. At some point, this will include Microsoft Visio, which is useful for coursework.

Expectations

Lecturers Expectations of Students

- Turn up for lectures (from next week, the content taught in the lectures will be used in the following weeks practical sessions)
- Arrive on time (there is usually useful information given out at the beginning of sessions)
- Participate and take notes in sessions
- Catch up on sessions if you miss them
- Finish the practical work before the following weeks practical sessions
- Study for about 4 hours a week total

These things are proven to increase the likelihood that a student gets a better mark at the end of the year.

Students Expectations of Lecturers

They are nice to students; start and end sessions on time; provide students with support and feedback on work throughout the module; and to return feedback and marks on work as quickly as they can (this usually should be within two weeks).

Assessments

There are two forms of assessment in this module.

Coursework

This will be worth 50% of the overall module mark. It will be released in the next few weeks and will be due at the end of the first week after the January assessment period (probably the Friday of that week at 11pm). The content assessed will all be from the first teaching block. We will get extra marks if we include content which hasn't been taught yet.

Exam

This will be worth 50% of the overall module mark. It will take place in the May/June assessment period and be computer based. It can include anything from the entire year however we won't have to write code (probably will have to look at code and say whats wrong). It will be multiple choice questions. There will be quizzes available on Moodle which will be similar to this where we can practice.

Brief Introduction to Databases

Database

"A single, possiblely large, repository of data that can be used simultaneuously by many departments and users" *(Database Solutions: A Step by Step Guide to Databases - T Connolly & C Begg)*

Spreadsheets

Spreadsheets are not databases. This is because a spreadsheet cannot hold the amount of data which a database can and eventhough though using some software, a database could be shared with multiple people, it cannot be edited by multiple people simultaneously.
This also applies to Microsoft Access.

Database Management System (DBMS)

DBMS

"The software which interacts with the users' application programs and the database" *(Database Solutions: A Step by Step Guide to Databases - T Connolly & C Begg)*

Examples of a DBMS include PostgreSQL, MySQL, SQL Server, Oracle and Mongo DB.

Why Use a Database

An alternative to databases are file based systems.
File based systems: are old fashioned; are not necessarily digital; they often contain duplicate data; are difficult to search; are very difficult to update; have the possibility to contain different file types which may not be compatible together; are inaccessible; and security may be an issue.
A database is: a modern approach; digital; duplicates can be removed; easy to search; easy to update; comprised of only one file type; capable of having multiple levels of access control; able to limit user access.
There are times at which a Database is not suitable for the setting. In this case, it may be more suitable to use a spreadsheet.

Integrated Database Environment

In an integrated database environment, the DBMS sites as a communication hub between all nodes. The DBMS is the server on which the database is hosted.
When the database is setup correctly, you can get more information out of it than you put in.

S.2. Practical 1

29-09-22 14:00 Mark & team FTC 3rd floor

Introduction to Practical sessions

Practical documents are available on Moodle, make a copy of these and store within your university Google Drive so you can edit them during the sessions and make notes.

Access Levels

In PostgreSQL, the first level of security is that a user cannot login unless they have been given access or there is a database with the same name as their username.
We don't have `sudo` access to linux, however we have full administrative access to PostgreSQL. Don't drop the database called `upxxxxxxx` (where `xxxxxxx` is replaced with student number) or anything that is owned by `postgres` as this breaks things.

PostgreSQL

PostgreSQL is ready to accept code when the prompt ends in `=#`. If you enter part of a command and press enter, the prompt will change to `-#`, this indicates that Postgres is waiting for you to finish the command.
PostgreSQL gives some useful error messages, SQL does not.

Code Editors

A code editor should be used to write SQL into, then the SQL should be copied and pasted into the Linux machine. The only thing that should be directly entered into the shell is to connect to a different database.
This is so that a. we have a copy of what we have done and b. so that if the VM is deleted, we are able to re-build our VM with less pain than if we didn't save all the code.
A recommended setup is to use VS code, with a SQL syntax extension. VS Code comes with integrated Powershell, allowing you to ssh to the VM from the same window.

SQL

SQL works like a procedural programming language, in that it reads the code inputted line by line. This also means that long and complex lines of code can be split across many lines, making it easier to read them.

Installing The First Database

Due to an issue with the image used to build the Virtual Machines, we have to create the database which we will use for the first few sessions. The code to do this was available on Moodle, copy and paste into the code editor then copy and paste again, this time into the Postgres prompt of the linux machine. This executes and creates the database, pre-populated with some sample data.

Tasks

1. List the databases in your server

LANGUAGE: SQL

```
\l
```

LANGUAGE: Unknown

```
                                  List of databases
    Name      |     Owner      | Encoding | Collate |  Ctype  |   Access privileges
--------------+----------------+----------+---------+---------+-----------------------
dsd_22        | up2108121      | UTF8     | C.UTF-8 | C.UTF-8 |
mongo-2021-fix | mongo-2021-fix | UTF8     | C.UTF-8 | C.UTF-8 |
postgres      | postgres       | UTF8     | C.UTF-8 | C.UTF-8 |
template0     | postgres       | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
              |                |          |         |         | postgres=CTc/postgres
template1     | postgres       | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
              |                |          |         |         | postgres=CTc/postgres
up2108121     | up2108121      | UTF8     | C.UTF-8 | C.UTF-8 |
(6 rows)
```

2. Connect to the database

LANGUAGE: SQL

```
\c dsd_22
```

LANGUAGE: Unknown

```
You are now connected to database "dsd_22" as user "up2108121".
```

3. List everything in this database

LANGUAGE: SQL

```
\d
```

LANGUAGE: Unknown

```
                    List of relations
Schema |            Name             |   Type   |   Owner
--------+-----------------------------+----------+-----------
public | category                    | table    | up2108121
public | category_cat_id_seq         | sequence | up2108121
public | cust_order                  | table    | up2108121
public | cust_order_cust_ord_id_seq  | sequence | up2108121
public | customer                    | table    | up2108121
public | customer_cust_id_seq        | sequence | up2108121
public | manifest                    | table    | up2108121
public | manifest_manifest_id_seq    | sequence | up2108121
public | product                     | table    | up2108121
public | product_prod_id_seq         | sequence | up2108121
public | role                        | table    | up2108121
public | role_role_id_seq            | sequence | up2108121
public | staff                       | table    | up2108121
public | staff_staff_id_seq          | sequence | up2108121
(14 rows)
```

4. List just the tables

LANGUAGE: SQL

```
\dt
```

LANGUAGE: Unknown

```
           List of relations
Schema |    Name    | Type  |   Owner
--------+------------+-------+-----------
public | category   | table | up2108121
public | cust_order | table | up2108121
public | customer   | table | up2108121
public | manifest   | table | up2108121
public | product    | table | up2108121
public | role       | table | up2108121
public | staff      | table | up2108121
(7 rows)
```

5. Get a list of all the customers in the `customer` table

LANGUAGE: SQL

```
SELECT * FROM customer;
```

LANGUAGE: Unknown

```
cust_id |   cust_fname    |    cust_lname    |        addr1          |    addr2     |      town
    ↪      | postcode  |              email
---------+-----------------+------------------+-----------------------+--------------+
    ↪ ----------------+-----------+---------------------------------
        1 | Jobey           | Boeter           | 6 Claremont Park      | Truax        | La
    ↪ Mohammedia  | CV42 3EF  | jboeter0@mail.ru
        2 | York            | O'Deegan         | 882 Hooker Trail      |              | Chemnitz
    ↪        | YA92 2OJ  | yodeegan1@nydailynews.com
        3 | Penelope        | Hexter           | 25 Jackson Lane       |              | Pingshan
    ↪        | LY32 8LN  | phexter2@cbslocal.com
        4 | Chadd           | Franz-Schoninger | 7 Division Point      | Texas        | Baojia
    ↪        | XA22 0UR  | cfranzschoninger3@google.com.hk
        5 | Vikky           | Eke              | 293 Colorado Drive    | Browning     | Kamenny
    ↪ Privoz | WQ12 3SF  | veke4@elegantthemes.com
        6 | Marie-francoise | Currier          | 032 Eagan Junction    | Duke         |
    ↪ Waekolong      | NB52 4MV  | acurrier0@economist.com
        7 | Benedicte       | Dozdill          | 579 Dryden Terrace    |              | Dawuhan
    ↪        | GY32 6GQ  | cdozdill1@amazon.de
        8 | Gorel           | Douthwaite       | 2946 Bluejay Parkway | Heath        | Sunbu
    ↪        | PH02 3ZX  | edouthwaite2@feedburner.com
        9 | Berengere       | Menendez         | 06154 Jackson Way     | Doe Crossing |
    ↪ Tsagaanders    | HO82 5XL  | amenendez3@dell.com
        10 | Pelagie         | Hachard          | 1777 Hauk Center      |              | Jiantou
    ↪         | NA52 4LM  | fhachard4@blinklist.com
        11 | Adaobi          | Musa             | 6 Clariss Ave         |              | La
    ↪ Mohammedia  | CV4 3F    | amusa9@mail.ca
(11 rows)
```

6. Choose a different table from the output of `\dt` and get a list of all the records in that table.

LANGUAGE: SQL

```
SELECT * FROM role;
```

LANGUAGE: Unknown

```
role_id |    role_name
---------+-----------------
   1 | Order Picker
    2 | Final Packer
    3 | Post Sales
    4 | Customer Retain
    5 | Misc
(5 rows)
```

S.3. The Database Environment

06-10-22 13:00 Mark RB LT1

Data or Information

When we think about real world things, we will generally think of these in terms of information, not data. Everyone and everything has information. We have to break information down into data to be able to store it.

Data

Facts and statistics collected together for reference or analysis
(https://en.oxforddictionaries.com/definition/data)

Information

The result of applying data processing to data, giving it context and meaning. Information can then be further processed to yield knowledge *(http://foldoc.org/information)*

When we need to store information in a database, we first have to break it down into data items. These can be entered into the database then pulled out again in different states. When done right, these different states should be able to tell us more information than we put in.
We also have knowledge, this is the ability to find things.

Processing Data

If we are given random data items, we can assume what they mean. For example, if we are given `1.99`; `cheeseburger`; and `Bob's Midnight Burgers`, you could assume that you could purchase a cheeseburger from an establishment called Bob's Midnight Burger for £1.99. However, this might be completely wrong! It could in fact be three un-related pieces of information or we may have mis-interpreted the information completely. This shows that it is imperative we look at the context which surrounds data, before drawing information from it.

Database Management System

The Database Management System (DBMS) is the core of the database system. Every communication to the database is done through the DBMS, this includes queries, data in and data out. The DBMS also controls access to the data and schema (which is stored within the database itself).

Schema

The 'blueprint' of the database.

An advantage of using a DBMS is that different users can be restricted as to what they can access; the data can easily be managed and the DBMS provides an integrated view of an enterprise's operations. The DBMS also removes the risk of inconsistent data and improves the ease with security can be controlled.

Database Languages

There are two different types of database languages (DDL and DML), each have a different purpose. SQL is both.

Before we look at DDL and DML in more detail, we first need to understand what the term 'Query' means.

Queries

A query is the code which interacts with the database.
This can be to read the contents of the database, you can 'query the database'. However it is also the code that puts the data into the database and the code which is used to build the database in the first place.

DDL

Data Definition Language (DDL) allows the DBA or users to describe and name entities, attributes and relationships required for the applications that access it and associated integrity and security constraints. It is the set of commands which are used to define the structure of the database. These are the commands used to create, modify or remove database objects (e.g., tables, users and indexes). Listed below are a number of the most commonly used DDL commands.

LANGUAGE: SQL

```sql
CREATE DATABASE
CREATE TABLE
ALTER TABLE
DROP DATABASE
DROP TABLE
RENAME TABLE
```

The following is an example of SQL code which creates a new table and as part of that defines the attributes within it.

LANGUAGE: SQL

```sql
create table property_for_rent (
    Property_id varchar(4) PRIMARY KEY,
    Street varchar(14) not null,
    City varchar(10) not null,
    Postcode varchar(10) not null,
    Type varchar(6) not null,
    Rooms integer not null,
    Rent decimal(6,2) not null,
    Owner_id varchar(4) not null REFERENCES private_owner(owner_id),
    Staff_id varchar(4) REFERENCES staff(Staff_id),
    branch_id varchar(4) REFERENCES branch(Branch_id)
);
```

DML

Data Manipulation Language (DML) provides the ability to manipulate data within the database. Its commands are used to select, insert, update and delete data items within a database. Listed below are a number of the most commonly used DML commands.
When selecting attributes to display, do not use `SELECT * FROM ...` as this selects everything. Instead, use `SELECT attribute, anotherAttribute, yetAnotherAttribute FROM ...`.
Take care when entering commands, for the configuration of our Virtual Machines, we are super users within PostgreSQL. Whatever we enter will be executed without question by the machine, this includes dropping data.

LANGUAGE: SQL

```sql
DELETE
INSERT
REPLACE
SELECT
```

```
5 UPDATE
```

The following is an example of SQL code which queries a table based on an attribute.

LANGUAGE: SQL

```sql
select property_id,
street,
city,
postcode,
owner_id from property_for_rent
where city = 'Glasgow';
```

S.4. Practical 2

06-10-22 14:00 Mark & team FTC Floor 3

Introductory Tasks

1. After getting into PostgreSQL client, list the databases.

LANGUAGE: SQL

```
\l
```

LANGUAGE: Unknown

```
    List of databases
    Name       |     Owner      | Encoding | Collate |  Ctype  |   Access privileges
---------------+----------------+----------+---------+---------+-----------------------
dsd_22         | up2108121      | UTF8     | C.UTF-8 | C.UTF-8 |
mongo-2021-fix | mongo-2021-fix | UTF8     | C.UTF-8 | C.UTF-8 |
postgres       | postgres       | UTF8     | C.UTF-8 | C.UTF-8 |
template0      | postgres       | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
               |                |          |         |         | postgres=CTc/postgres
template1      | postgres       | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
               |                |          |         |         | postgres=CTc/postgres
up2108121      | up2108121      | UTF8     | C.UTF-8 | C.UTF-8 |
(6 rows)
```

2. Connect to the `dsd_22` database.

LANGUAGE: SQL

```
\c dsd_22
```

LANGUAGE: Unknown

```
You are now connected to database "dsd_22" as user "up2108121".
```

3. List the contents of the database

LANGUAGE: SQL

```
\d
```

LANGUAGE: Unknown

```
                    List of relations
Schema |             Name             |   Type   |   Owner
--------+------------------------------+----------+-----------
public | category                     | table    | up2108121
public | category_cat_id_seq          | sequence | up2108121
public | cust_order                   | table    | up2108121
public | cust_order_cust_ord_id_seq   | sequence | up2108121
public | customer                     | table    | up2108121
public | customer_cust_id_seq         | sequence | up2108121
public | manifest                     | table    | up2108121
public | manifest_manifest_id_seq     | sequence | up2108121
public | product                      | table    | up2108121
public | product_prod_id_seq          | sequence | up2108121
public | role                         | table    | up2108121
public | role_role_id_seq             | sequence | up2108121
```

```
public | staff                      | table    | up2108121
public | staff_staff_id_seq         | sequence | up2108121
(14 rows)
```

5. List just the tables

LANGUAGE: SQL

```
\dt
```

LANGUAGE: Unknown

```
            List of relations
Schema |    Name     | Type  |   Owner
--------+-------------+-------+-----------
public | category    | table | up2108121
public | cust_order  | table | up2108121
public | customer    | table | up2108121
public | manifest    | table | up2108121
public | product     | table | up2108121
public | role        | table | up2108121
public | staff       | table | up2108121
(7 rows)
```

The `\dt` command removes the sequences (which will be discussed further in a couple of weeks time).
6. Look at the structure of the role table.

LANGUAGE: SQL

```
\d role
```

LANGUAGE: Unknown

```
Table "public.role"
Column    |          Type          | Collation | Nullable |                 Default
----------+------------------------+-----------+----------+-----------------------------------------
    ↪
role_id   | integer                |           | not null | nextval('role_role_id_seq'::regclass
    ↪ )
role_name | character varying(20) |           |          |
Indexes:
    "role_pkey" PRIMARY KEY, btree (role_id)
Referenced by:
    TABLE "staff" CONSTRAINT "staff_role_fkey" FOREIGN KEY (role) REFERENCES role(role_id)
```

Using SQL To Access Data

Most of the commands used so far are PostgreSQL specific commands (these are the ones which begin with \).
If the output from a command is too long, PostgreSQL will show a colon (`:`) at the bottom of the screen. To show the next screen, press the space bar. Once all the records have been seen, the screen will show `(END)`. At this point, hit `q` to exit back to the prompt. `q` can also be pressed at the colon to exit back to the prompt from there too.
1. Read all the records in the `dsd_22` table `category`.

LANGUAGE: SQL

```
SELECT * FROM CATEGORY;
```

LANGUAGE: Unknown

```
cat_id |  cat_name
--------+-------------
     1 | Men's Wear
     2 | Ladies Wear
     3 | Kid's Wear
     4 | Outdoor
     5 | Sport
     6 | Health
(6 rows)
```

2. Run the following command and see if the output is different.

LANGUAGE: SQL

```
select * from category;
```

LANGUAGE: Unknown

```
cat_id |  cat_name
--------+-------------
     1 | Men's Wear
     2 | Ladies Wear
     3 | Kid's Wear
     4 | Outdoor
     5 | Sport
     6 | Health
(6 rows)
```

3. Run the following command, and see if the output is different.

LANGUAGE: SQL

```
select * from 'Category';
```

LANGUAGE: Unknown

```
ERROR:  syntax error at or near "'Category'"
LINE 1: select * from 'Category';
                      ^
```

4. Run the following command and see if the output is different.

LANGUAGE: SQL

```
select * from "Category";
```

LANGUAGE: Unknown

```
ERROR:  relation "Category" does not exist
LINE 1: select * from "Category";
                      ^
```

5. Run the following command and see if the output is different.

LANGUAGE: SQL

```
select * from 'category';
```

```
LANGUAGE: Unknown
ERROR:  syntax error at or near "'category'"
LINE 1: select * from 'category';
                      ^
```

6. Run the following command and see if the output is different.

```
LANGUAGE: SQL
select * from "category";
```

```
LANGUAGE: Unknown
cat_id |  cat_name
--------+-------------
     1 | Men's Wear
     2 | Ladies Wear
     3 | Kid's Wear
     4 | Outdoor
     5 | Sport
     6 | Health
(6 rows)
```

7. Run the `\dt` command again, look at the case of the table names.
8. Run the following command (nb, this is supposed to contain non-standard quote marks as copied from the Google Doc).

```
LANGUAGE: SQL
SELECT * FROM "category";
```

```
LANGUAGE: Unknown
ERROR:  relation ""category"" does not exist
LINE 1: SELECT * FROM "category";
```

From these exercises, it is clear that case doesn't matter when the table name is not in quotes; and that the type of quotes used matter (there are extensions available for Google Docs which allow code to be stored in them and for it to keep its formatting).

Table Structure

To see how tables are linked together, it is possible to view the table structures. This information tells you how the attributes are linked together and what the data types and sizes of said data types are (where this is applicable).
1. Run the following command.

```
LANGUAGE: SQL
\d customer
```

```
LANGUAGE: Unknown
    Table "public.customer"
    Column   |           Type           | Collation | Nullable |                  Default

    ↪ ------------+------------------------+-----------+----------+-------------------------------------------
```

```
    ↪
 cust_id    | integer                |           | not null | nextval('customer_cust_id_seq'::
   ↪ regclass)
 cust_fname | character varying(25)  |           | not null |
 cust_lname | character varying(35)  |           | not null |
 addr1      | character varying(50)  |           | not null |
 addr2      | character varying(50)  |           |          |
 town       | character varying(60)  |           | not null |
 postcode   | character(9)           |           | not null |
 email      | character varying(255) |           | not null |
Indexes:
    "customer_pkey" PRIMARY KEY, btree (cust_id)
Referenced by:
    TABLE "cust_order" CONSTRAINT "cust_order_cust_id_fkey" FOREIGN KEY (cust_id) REFERENCES
   ↪ customer(cust_id)
```

From the output, we can see that the data type of `cust_id` is integer and the data type of `postcode` is a fixed 9 length character.

Creating new Tables in SQL

The syntax for creating a table (or relation, if we're being proper) is shown below.

LANGUAGE: SQL

```sql
CREATE TABLE tableName(
attributeName dataType (options),
attributeName dataType (options),
...
);
```

Task

1. Create a new database with a name of your choice.
2. Connect to the database.
3. Create a new table with two attributes (one of data type INT, that is also the primary key and one that has a data type of your own choosing).

LANGUAGE: SQL

```sql
CREATE DATABASE week02;

CREATE TABLE NEWTABLE(
IAMNUMBER INT PRIMARY KEY,
IAMSTRING VARCHAR(10)
);
```

Now, insert a record into the table.

LANGUAGE: SQL

```sql
INSERT INTO NEWTABLE (IAMNUMBER, IAMSTRING) VALUES(12, 'cheese');
```

Now, insert another new record into the table, using the same INT value as the first record. Take note of the message which is displayed.

LANGUAGE: SQL

```sql
INSERT INTO NEWTABLE (IAMNUMBER, IAMSTRING) VALUES(12, 'ham');
```

LANGUAGE: Unknown

```
ERROR:  duplicate key value violates unique constraint "newtable_pkey"
DETAIL:  Key (iamnumber)=(12) already exists.
```

S.5. Database Concepts

13-10-22 13:00 Mark RB LT1

Despite the fact that the relational database model was designed by Codd in the 1970s, it is a valid system and used widely.

Key Terms

Database Term	Description
Entity	An object or a 'thing' about which data is stored.
Attributes	Some quality associated with the entity (eg ID number, username, size). These have data types (eg number, string etc) and maximum sizes. Other terms are elements and properties.
Relation	A two dimensional representation (table) of entities and/ or relationships. Other terms used are relation table or table.
Entity Set	A set of entities of the same type.
Relationship	How two relations (tables) are related to each other. Relationships are represented in relations.
Tuple	Corresponds to rows of the table or records of a relation. Other terms used are record and row.
Domain	A pool of all legal values from which actual attribute values are drawn.
Primary Key	An attribute or combination of attributes for which values uniquely identify tuples in the relation. The primary key is chosen from a set of candidate keys. If you have a numeric value which the system can generate, let it do it for you.
Candidate Key	There may be more than one potential primary keys for a relation. Each is called a candidate key or super-key.
Alternate Key	An alternate access path to data that is not via the primary key.
Composite Key	A combination of attributes that act as a candidate key in a relation. Each participating attribute in the composite key (also known as candidate key) is called a simple key.
Foreign Key	An attribute (or combination of attributes) that is a primary key in another relation. They can appear many times.
Degree	Number of attributes in a relation; also called the arity.

When designing a database, the first thing you need to think about is what entities do you need to store information about. Then think about the attributes which you need to store about each entity. Then create relations. At this point, think about the domain for any of the attributes (for example, month 1-12 or day 0-6 (Sunday to Saturday) or hours 0-23). Now think about keys.

Entity

An entity is a thing, it could be a person or a specific type of person.
To identify entities, look at the information given to you and identify the nouns. The nouns give an idea of what the entities look like but they require fine tuning.
There can be as many entities as needed.
We can describe entities using their attributes.

We now think about keys.

Primary Key

To identify what will be a primary key, we look for something that is unique. This should be something which cannot be changed. If there is nothing suitable, create your own primary key.

Foreign key

Does not have to be primary key in other table, however it has to be unique within the other table.

S.6. Practical 03

13-10-22 14:00 Mark and team FTC Floor 3

Q1. using the `count()` function demonstrated by your tutor, how many records are there in each of the tables in the dsd_22 database. (Remember to use `\dt` to give you a list of tables in the database.) Copy the outputs below.

LANGUAGE: Unknown

```
dsd_22=# select count(*) from category;
count
-------
    6
(1 row)
dsd_22=# select count(*) from cust_order;
count
-------
150
(1 row)
dsd_22=# select count(*) from customer;
count
-------
11
(1 row)
dsd_22=# select count(*) from manifest;
count
-------
150
(1 row)

dsd_22=# select count(*) from product;
count
-------
100
(1 row)

dsd_22=# select count(*) from role;
count
-------
    5
(1 row)

dsd_22=# select count(*) from staff;
count
-------
10
(1 row)
```

Q2. Use the max() function to find the highest value of the role_id attribute in the role table. Copy the output below

LANGUAGE: SQL

```sql
select max(role_id) from role;
```

LANGUAGE: Unknown

```
max
-----
    5
(1 row)
```

Q3. Insert a new row of data into the role table with

LANGUAGE: SQL

```sql
INSERT INTO ROLE (ROLE_NAME) VALUES ('Pre Sales');
```

Q4. How many rows of data are now in the role table? Copy it below.

LANGUAGE: SQL

```sql
select count(*) from role;
```

LANGUAGE: Unknown

```
count
-------
    6
(1 row)
```

Q5. What is the maximum value of the role_id now? Copy it below.

LANGUAGE: SQL

```sql
select max(role_id) from role;
```

LANGUAGE: Unknown

```
max
-----
    6
(1 row)
```

Q6. Delete this new row with

LANGUAGE: SQL

```sql
DELETE FROM ROLE WHERE ROLE_NAME = 'Pre Sales';
```

LANGUAGE: Unknown

```
DELETE 1
```

Q7. How many rows of data are now in the role table? Copy it below.

LANGUAGE: Unknown

```
count
-------
      5
(1 row)
```

Q8. What is the maximum value of the role_id now? Copy it below.

LANGUAGE: Unknown

```
max
-----
    5
(1 row)
```

Q9. Reinsert the row of data into the role table again with

LANGUAGE: SQL

```sql
INSERT INTO ROLE (ROLE_NAME) VALUES ('Cleaning Team');
```

LANGUAGE: Unknown

```
INSERT 0 1
```

Q10. How many rows of data are now in the role table? Copy it below.

LANGUAGE: Unknown

```
count
-------
        6
(1 row)
```

Q11. What is the maximum value of the role_id now? Copy it below.

LANGUAGE: Unknown

```
max
-----
    6
(1 row)
```

Q12. Create a random value using the random function. Copy the value below

LANGUAGE: SQL

```sql
SELECT RANDOM();
```

LANGUAGE: Unknown

```
      random
-------------------
 0.175315219908953
(1 row)
```

Q12a. Create another random number. Copy the value below

LANGUAGE: SQL

```sql
SELECT RANDOM();
```

LANGUAGE: Unknown

```
      random
-------------------
 0.272884896956384
(1 row)
```

Q13. Create one more random value but now multiply it by 11. Remember that to multiply you do not use x but use the * symbol. Run this code 5 times and copy the values below.

LANGUAGE: Unknown

```
dsd_22=# select random()*11;
     ?column?
------------------
```

```
 9.60335403773934
(1 row)
dsd_22=# select random()*11;
     ?column?
------------------
 0.160588529892266
(1 row)

dsd_22=# select random()*11;
     ?column?
------------------
 5.25661591161042
(1 row)

dsd_22=# select random()*11;
     ?column?
------------------
 7.78145408304408
(1 row)
dsd_22=# select random()*11;
     ?column?
------------------
 10.1819118564017
(1 row)
```

Q14. Connect to your home database, upxxxxxxx and run the following code to create a new table and insert some random numbers into it.

LANGUAGE: SQL

```sql
create table numb1(numb_id int primary key, ran_val decimal(17,15));

insert into numb1(numb_id, ran_val) values
(1,random()),(2,random()),(3,random()),(4,random()),(5,random()),(6,random()),(7,random()),(8,
    ↪ random()),(9,random()),(10,random());
```

LANGUAGE: Unknown

```
    INSERT 0 10
```

Q14a. Check that there are 10 rows of data with `SELECT COUNT(*) FROM NUMB1;` If not, check your output for any error messages. You should get responses below except the prompt will be your student id number.

LANGUAGE: Unknown

```
 count
-------
    10
(1 row)
```

Q15. Run a `SELECT * FROM NUMB1;` Copy the output below.

LANGUAGE: Unknown

```
 numb_id |      ran_val
---------+-------------------
       1 | 0.481754711363465
       2 | 0.020102311857045
       3 | 0.541421711910516
       4 | 0.046512784436345
       5 | 0.842869907151908
       6 | 0.137599688488990
       7 | 0.925696460530162
       8 | 0.765472991392016
       9 | 0.712954005226493
      10 | 0.161490791942924
```

```
(10 rows)
```

Q15a. Compare the values that you get with the values below. They should be different. This is because the code used inserts a fixed value, the numb_id and a completely random value into the ran_val attribute for each row.

LANGUAGE: Unknown

```
test_num=# SELECT * FROM NUMB1;
 numb_id |      ran_val
---------+-------------------
       1 | 0.477631121408194
       2 | 0.978080025874078
       3 | 0.516494689509273
       4 | 0.849129045847803
       5 | 0.484937957022339
       6 | 0.895700289402157
       7 | 0.852438564877957
       8 | 0.727535046637058
       9 | 0.062769805546850
      10 | 0.594313766807318
(10 rows)
```

Q16. Find the highest value of ran_val using the max() function. Copy it below.

LANGUAGE: SQL

```
select max(ran_val) from numb1;
```

LANGUAGE: Unknown

```
        max
-------------------
 0.925696460530162
(1 row)
```

Q17. Find the lowest value of ran_val using the min() function. Copy it below.

LANGUAGE: SQL

```
select min(ran_val) from numb1;
```

LANGUAGE: Unknown

```
        min
-------------------
 0.020102311857045
(1 row)
```

Q18. What is the average value of ran_val. Reminder: look at the basic functions document for ideas.

LANGUAGE: SQL

```
select avg(ran_val) from numb1;
```

LANGUAGE: Unknown

```
          avg
------------------------
```

```
 0.46358753642998640000
(1 row)
```

Q19. What is the current timestamp on your server? Copy it below

LANGUAGE: SQL

```sql
select now();
```

LANGUAGE: Unknown

```
              now
-------------------------------
 2022-10-13 13:43:49.196518+00
(1 row)
```

Q20. What is the first name of the customer with the ID number of 3?

LANGUAGE: SQL

```sql
select cust_fname from customer where cust_id=3;
```

LANGUAGE: Unknown

```
 cust_fname
------------
 Penelope
(1 row)
```

Q21. What is the category id number of the outdoor category? Copy below.

LANGUAGE: SQL

```sql
select cat_id from category where cat_name='Outdoor';
```

LANGUAGE: Unknown

```
 cat_id
--------
      4
(1 row)
```

Q22. How many orders in the cust_order table are for cust_id 15? Copy below.

LANGUAGE: SQL

```sql
select count(*) from cust_order where cust_id=15;
```

LANGUAGE: Unknown

```
 count
-------
     0
(1 row)
```

Q23. List the first and last names of the staff members who live in Portsmouth. Copy below.

LANGUAGE: SQL

```sql
select staff_fname, staff_lname from staff where town='Portsmouth';
```

LANGUAGE: Unknown

```
 staff_fname | staff_lname
-------------+-------------
 Niel        | Welsby
 Janeva      | Gillicuddy
(2 rows)
```

Q24. What values does addr1 and addr2 have for the staff member whose id = 4? Copy below.

LANGUAGE: SQL

```sql
    select addr1 , addr2 from staff where staff_id=4;
```

LANGUAGE: Unknown

```
      addr1       | addr2
------------------+-------
 959 Algoma Plaza |
(1 row)
```

Q25. How many members of staff have the role value of 3? Copy below.

LANGUAGE: SQL

```sql
select count(*) from staff where role=3;
```

LANGUAGE: Unknown

```
 count
-------
     3
(1 row)
```

Q26. How many products are in the product category = 2?

LANGUAGE: SQL

```sql
select count(*) from product where prod_id=2;
```

LANGUAGE: Unknown

```
 count
-------
     1
(1 row)
```

S.7. Coursework & Entity Relationship Diagrams

20-10-22 13:00 Mark RB LT1

Coursework

Coursework

The coursework is now available on Moodle, within Assessment and Support Materials.

The deadline for the coursework isn't until February.
It is recommended to submit the files to Moodle well in advance of the deadline because there is a chance there will be a technical issue with Moodle when the deadline is, no extenuating circumstances will be given if this is the case.
The Entity Relationship Diagram submitted must be produced digitally, hand drawn diagrams will gain 0 credits.
Mark uses Mocakroo and Lucid Charts for generating dummy data and drawing ERDs respectively. This is what works well for him, there are other platforms available for both, with more information in the Coursework document.

Entity Relationship Diagrams

Entity Relationship Diagrams (ERDs) are diagrams which show how entities are related, down to the detail of what the attributes are and how they relate to each other as well.

Business Rules

When designing databases, business rules will be taken into consideration.

Business Ruls

A statement that defines how a company does stuff or how stuff works within a company.

We can use business rules to help guide us on how to design databases.

Relationship Links

We will be using Crows Foot Notation, there are a number of other types of notation however we won't look at any of these.

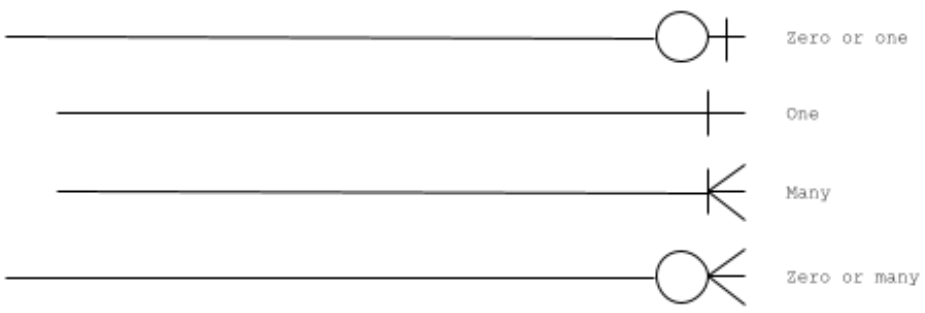

Figure 1: Crow's feet notation

When designing entities, it is important to name them in the singular, for example `pig` not `pigs`, and to use underscore notation where multiple words comprise the entity name, not camel notation.
Many-to-many relationships are not permitted. We will return to this in a future lecture.

Constraints

> **Constraint**
>
> A rule that protects your data or enforces certain behaviour.

For example, a constraint may be set to be `NOT NULL`, this would ensure that whenever a row of data is inserted into a table, that attribute would have to contain data.
Keys are constraints. The primary key is automatically set to be `NOT NULL`, we do not have to specify that when creating a table. We could use a default constraint, to specify the the time that a record was entered into a table.
Check constraints can be used to validate data as it is entered, for example a price must contain two decimal places. Check may be needed as part of the coursework.

S.8. Practical: SQL and Entities

20-10-22 14:00 Mark & Team FTC 3rd Floor

Task 1: Run the provided code and observe the outputs.

Run the following DDL code.

LANGUAGE: SQL

```sql
CREATE DATABASE customer_db;

\c customer_db

CREATE TABLE customer1 (cust_id SERIAL PRIMARY KEY, cust_fname VARCHAR(20) NOT NULL, cust_lname
    ↪  VARCHAR(20) NOT NULL);

\d customer1

ALTER TABLE customer1 ADD COLUMN cust_email varchar(100) NOT NULL UNIQUE;

\d customer1

DROP TABLE customer1;

-- check that the table is gone now

-- anything in the line after the two dashes is a comment by the way

\l
\d customer1
```

Run the following DML code.
Creating a new table and populating it with some dummy data.

LANGUAGE: SQL

```sql
CREATE TABLE customer (cust_id SERIAL PRIMARY KEY, cust_fname VARCHAR(20) NOT NULL, cust_lname
    ↪ VARCHAR(20) NOT NULL, cust_email varchar(60) NOT NULL);

INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (22,'Kamil', 'Novak',
    ↪ 'kamnovak@gmail.com');

INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (66,'Aarav', 'Anand',
    ↪ 'aanand98@gmail.com');

INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (67,'Alia', 'Anand','
    ↪ aanand98@gmail.com');
```

Viewing what is in the table

LANGUAGE: SQL

```sql
SELECT * FROM customer;

SELECT cust_fname, cust_email from customer;
```

Selecting only the attributes which we need, so we don't have to retrieve all of the data from a table.

LANGUAGE: SQL

```sql
SELECT cust_email, cust_id, cust_fname, cust_lname from customer;
```

Insert more records, some of these return errors.

LANGUAGE: SQL

```sql
UPDATE customer SET cust_email = 'i_love_elephants_1@gmail.com' where cust_id = 66;
SELECT * FROM customer;

INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (67,'Alia', 'Anand','aanand98@gmail.com');

INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (70,'Connor', 'Murphy','connormurphy99@gmail.com');

INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (71,'Connor', 'Murphy','connormurphy199@gmail.com');

INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (1,'Tim', 'Nice-but-Dimm','alongemailaddresswillfit@quitealongdomainnametoo.com');

INSERT INTO customer ( cust_fname, cust_lname, cust_email) VALUES ('Tim', 'Nice-but-Dimm','averylongemailaddresswillnotfit1111111111111111111111111111111111111111111a@quitealongdomainnametoo.com');
```

Task 2: Write SQL code for the following questions.

1. Create a new database called `code_test`

LANGUAGE: SQL

```sql
CREATE DATABASE code_test;
```

2. Connect to this new database

LANGUAGE: SQL

```sql
\c code_test
```

3. Create a new table called `table_one`, which has the following attributes
   (a) `Record_id` an integer
   (b) `Att_1` a varchar that will hold upto 30 characters
   (c) `Att_2` a char that will hold 10 characters
   (d) `Att_3` a decimal that can hold the value of 9.99.

LANGUAGE: SQL

```sql
CREATE TABLE table_one(Record_id INT PRIMARY KEY, Att_1 VARCHAR(30), Att_2 CHAR(10), Att_3 DECIMAL(3,2));
```

4. Look at the structure of this table once you have created it. Show the output below.

LANGUAGE: SQL

```sql
\d table_one
```

LANGUAGE: Unknown

```
                    Table "public.table_one"
  Column   |         Type          | Collation | Nullable | Default
-----------+-----------------------+-----------+----------+---------
 record_id | integer               |           | not null |
 att_1     | character varying(30) |           |          |
 att_2     | character(10)         |           |          |
 att_3     | numeric(3,2)          |           |          |
```

```
Indexes:
    "table_one_pkey" PRIMARY KEY, btree (record_id)
```

5. Alter the table by adding a new column called `Att_4` that will hold another integer.

```
LANGUAGE: SQL
ALTER TABLE table_one ADD COLUMN Att_4 INT;
```

6. Look at the structure of this table again once you have added this new column. Show the output below.

```
LANGUAGE: SQL
\d table_one
```

```
LANGUAGE: Unknown
                      Table "public.table_one"
  Column   |         Type          | Collation | Nullable | Default
-----------+-----------------------+-----------+----------+---------
 record_id | integer               |           | not null |
 att_1     | character varying(30) |           |          |
 att_2     | character(10)         |           |          |
 att_3     | numeric(3,2)          |           |          |
 att_4     | integer               |           |          |
Indexes:
    "table_one_pkey" PRIMARY KEY, btree (record_id)
```

7. Insert two records into the table called `table_one`

   (a) Record_id = 1, Att_1 = continent , Att2 = 0olP$fguj , Att_3 = 9.99 , Att_4 = 42

   (b) Record_id = 2, Att_1 = Portsmouth University , Att2 = Violet , Att_3 = 9.99 , Att_4 = 99999

```
LANGUAGE: SQL
INSERT INTO table_one (Record_id, Att_1, Att_2, Att_3, Att_4) VALUES (1, 'continent', '0
    ↪ olP[dollarSign]fguj', 9.99, 42);
INSERT INTO table_one (Record_id, Att_1, Att_2, Att_3, Att_4) VALUES (2, 'Portsmouth
    ↪ University', 'Violet', 9.99, 9999);
```

8. Get all fo the data from the table

```
LANGUAGE: SQL
SELECT * FROM table_one;
```

9. Get a screenshot of the data

```
LANGUAGE: Unknown
 record_id |         att_1          |   att_2    | att_3 | att_4
-----------+------------------------+------------+-------+-------
         1 | continent              | 0olP[dollarSign]fguj  |  9.99 |    42
         2 | Portsmouth University | Violet     |  9.99 |  9999
(2 rows)
```

10. Change the value of `Att_4` in record 1 from `44` to `66`

```
LANGUAGE: SQL
UPDATE table_one SET Att_4 = 66 WHERE record_id = 1;
```

11. Get the data from the table for only record 1

```
LANGUAGE: SQL
SELECT * FROM table_one WHERE record_id = 1;
```

12. Get a screenshot of the results.

```
LANGUAGE: Unknown
 record_id |   att_1   |   att_2    | att_3 | att_4
-----------+-----------+------------+-------+-------
         1 | continent | 0olP[dollarSign]fguj  |  9.99 |    66
(1 row)
```

S.9. ERD, Attributes & Datatypes

27-10-22 13:00 RB LT1 Mark

Attributes

An entity is a thing. The attributes, of an entity, are the things which describe the thing. We need to be able to identify individual entities.

Example: People

If we are having a person as an entity, the attributes we will probably need are: date of birth; given name; family name. There are attributes which we don't need to store (for example: weight, height).

Addresses

When we store people, we will usually store their address in their record. This will be explored when do normalisation after consolidation week.

GDPR

When we store data, we have to be sure we are being GDPR compliant and storing what what you need to store.
GDPR states that you must ensure the personal data you are processing is:

- adequate - sufficient to properly fulfil your stated purpose;
- relevant - has a rational link to that purpose; and
- limited to what is necessary - you do not hold more than you need for that purpose.

Data Types

Now we know what attributes we need to store about the attribute, we need to think about types of data that is.

Names

Names are made up from characters, these could include apostrophes and hyphens. There is a question here as to how long names can be. A rule of thumb would be to use 20 characters for first name and 25 for surnames.

Numeric

There are a number of different numeric data types.

- `smallint` - holds an integer range -32768 to +32767
- `integer` - holds an integer range -2147483648 to +2147483647
- `bigint` - holds an integer range -9223372036854775808 to +9223372036854775807
- `decimal` - holds a decimal number with up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
- `real` - similar to decimal but provides 6 decimal digits precision

- `double` - similar to real but provides 15 decimal digits precision
- `serial` - holds an integer range 1 to 2147483647
- `bigserial` - holds an integer range 1 to 9223372036854775807

Characters

There are a number of different character data types.
Phone numbers should be stored as a character not as a numeric data type as they will often have leading zeros.

- `text` - variable 'unlimited' length
- `character/ char` - fixed length (blank padding is added if less than given size)
- `varying character / varchar` - variable length with limit

Dates and Times

There are a number of different date/ time data types.

- `timestamp without timezone` - both date and time (no time zone) range 4713 BC to 294276 AD with 1 microsecond resolution
- `timestamp with timezone` - both date and time (with time zone) range 4713 BC to 294276 AD with 1 microsecond resolution
- `date` - date without time range 4713 BC to 5874897 AD with 1 day resolution
- `time without timezone` - time of day (no date) range 00:00:00 to 24:00:00 with 1 microsecond resolution
- `time with timezone` - time of day (no date), with time zone range 00:00:00 to 24:00:00 with 1 microsecond resolution and adjustment for time zone

Example of drawing up an entity

If we have a draft entity with the following attributes `cust_id`, `cust_name`, `addeess`, `email`. This presents a number of problems.
If we want to search for a specific name, this is more complicated because the customer name is stored as a single attribute where it should be multiple attributes.
Addresses should not be stored as a single attribute.

Break down data

We should break down information into usable data. For example, addresses should be broken down into: `address1`, `address2`, `town`, `county`, `postcode`, `country`.
Names should be broken down into `firstName`, `lastName`. It could also be argued that a single middle name could also be included.

Adding data types

- `cust_id - int`
- `cust_fname - varchar`
- `cust_mname - varchar`

- `cust_lname` - `varchar`
- `addr1` - `varchar`
- `addr2` - `varchar`
- `town` - `varchar`
- `postcode` - `char` (could be a `varchar`)
- `email` - `varchar`

Sizes of data types

Now we have worked out what data types we want to use, we need to think about the sizes of those data types.

S.10. Normalisation

10-11-22 13:00 Mark RB LT1

Introduction to Normalisation

Normalisation is the process of designing a database in a way that reduces data redundancy and makes the database more efficient. As part of doing this, we have set rules to follow which enables us to decide what is stored in an entity and then within a table.
There are five levels of normalisation, information which has not been normalised is in zero form and a database that has been normalised will be in 3rd normal form.

First Normal Form

Rules for a table to be in 1NF:

- It should only have single (atomic) valued attributes/ columns (each column should not hold more than one value)
- Values stored in a column should be of the same domain (this means don't hold char data in one row and int in another, both in the same columns)
- All the columns in a table should have unique names (there cannot be two or more columns or attributes with the same name)
- The order in which data is stored doesn't matter

Whilst converting data to the first normal form, you may find that a new entity is created. This can be done to reduce data redundancy.

Second Normal Form

Rules for a table to be in 2NF:

- Be in 1NF
- Have no partial dependencies

A partial dependency is where part of an attribute can be identified by something other than the primary key.

Third Normal Form

Rules for a table to be in 3NF:

- Be in 2NF
- Not have transitive dependencies

A transitive dependency is a n attribute which is dependent on an attribute which is not the primary key.

S.11. Practical: Keys & Joins

10-11-22 14:00 Mark and team FTC Floor 3

1. Connect to the `dsd_22` Database
2. Drop the `dsd_22` database using the code shown below and show the output below.

```sql
DROP DATABASE dsd_22;
```

```
ERROR:  cannot drop the currently open database
```

3. If you were unable to drop the database, how did you do it? Show your code below.

```sql
\c up2108121
DROP DATABASE dsd_22;
```

```
DROPPED DATABASE
```

4. Create the table but do not create any tableofcontents

```sql
CREATE DATABASE dsd_22;
```

5. Exit Postgres client but don't close connection to the VM
6. Download the code from Moodle
7. Use SCP through the terminal to copy the file to the virtual machine
8. Run the code to populate the database
9. Connect to the `dsd_22` database.
10. Check that the tables have been created with the `\dt` command and to check that there is data in each of them, select the number of rows in each table.

```
SELECT COUNT(*) FROM category;
 count
-------
     6
(1 row)

SELECT COUNT(*) FROM cust_order;
 count
-------
   150
(1 row)

```

```
SELECT COUNT(*) FROM customer;
 count
-------
    11
(1 row)

SELECT COUNT(*) FROM manifest;
 count
-------
   150
(1 row)

SELECT COUNT(*) FROM product;
 count
-------
   100
(1 row)

SELECT COUNT(*) FROM role;
 count
-------
     5
(1 row)

SELECT COUNT(*) FROM staff;
 count
-------
    10
(1 row)
```

11. Get a printout of the structure of each table by using the `\d` command.

LANGUAGE: Unknown

```
\d category
    Table "public.category"
Column  |          Type          | Collation | Nullable |                 Default

----------+------------------------+-----------+----------+----------------------------------------
    ↪
cat_id   | integer                |           | not null | nextval('category_cat_id_seq'::
    ↪ regclass)
cat_name | character varying(40) |           |          |
Indexes:
"category_pkey" PRIMARY KEY, btree (cat_id)
Referenced by:
TABLE "product" CONSTRAINT "product_prod_cat_fkey" FOREIGN KEY (prod_cat) REFERENCES
    ↪ category(cat_id)

\d cust_order
 Table "public.cust_order"
Column    |  Type   | Collation | Nullable |                     Default
------------+---------+-----------+----------+-------------------------------------------------
    ↪
cust_ord_id | integer |           | not null | nextval('cust_order_cust_ord_id_seq'::
    ↪ regclass)
staff_id    | integer |           |          |
cust_id     | integer |           |          |
Indexes:
"cust_order_pkey" PRIMARY KEY, btree (cust_ord_id)
Foreign-key constraints:
"cust_order_cust_id_fkey" FOREIGN KEY (cust_id) REFERENCES customer(cust_id)
"cust_order_staff_id_fkey" FOREIGN KEY (staff_id) REFERENCES staff(staff_id)
Referenced by:
TABLE "manifest" CONSTRAINT "manifest_cust_ord_id_fkey" FOREIGN KEY (cust_ord_id)
    ↪ REFERENCES cust_order(cust_ord_id)

\d customer
      Table "public.customer"
Column   |          Type          | Collation | Nullable |                 Default

------------+------------------------+-----------+----------+----------------------------------------
    ↪
```

```
cust_id    | integer                |           | not null | nextval('customer_cust_id_seq
    ↪ '::regclass)
cust_fname | character varying(25)  |           | not null |
cust_lname | character varying(35)  |           | not null |
addr1      | character varying(50)  |           | not null |
addr2      | character varying(50)  |           |          |
town       | character varying(60)  |           | not null |
postcode   | character(9)           |           | not null |
email      | character varying(255) |           | not null |
Indexes:
"customer_pkey" PRIMARY KEY, btree (cust_id)
Referenced by:
TABLE "cust_order" CONSTRAINT "cust_order_cust_id_fkey" FOREIGN KEY (cust_id) REFERENCES
    ↪ customer(cust_id)

\d manifest
 Table "public.manifest"
Column     |  Type   | Collation | Nullable |                    Default
-------------+---------+-----------+----------+-----------------------------------------------
    ↪
manifest_id | integer |           | not null | nextval('manifest_manifest_id_seq'::
    ↪ regclass)
cust_ord_id | integer |           | not null |
prod_id     | integer |           | not null |
Indexes:
"manifest_pkey" PRIMARY KEY, btree (manifest_id)
Foreign-key constraints:
"manifest_cust_ord_id_fkey" FOREIGN KEY (cust_ord_id) REFERENCES cust_order(cust_ord_id)
"manifest_prod_id_fkey" FOREIGN KEY (prod_id) REFERENCES product(prod_id)


\d product
     Table "public.product"
Column   |         Type          | Collation | Nullable |                Default

-----------+-----------------------+-----------+----------+---------------------------------------
    ↪
prod_id   | integer               |           | not null | nextval('product_prod_id_seq'::
    ↪ regclass)
prod_name | character varying(50) |           | not null |
prod_cat  | integer               |           | not null |
Indexes:
"product_pkey" PRIMARY KEY, btree (prod_id)
Foreign-key constraints:
"product_prod_cat_fkey" FOREIGN KEY (prod_cat) REFERENCES category(cat_id)
Referenced by:
TABLE "manifest" CONSTRAINT "manifest_prod_id_fkey" FOREIGN KEY (prod_id) REFERENCES
    ↪ product(prod_id)

\d role
     Table "public.role"
Column   |         Type          | Collation | Nullable |                Default

-----------+-----------------------+-----------+----------+-------------------------------------
    ↪
role_id   | integer               |           | not null | nextval('role_role_id_seq'::
    ↪ regclass)
role_name | character varying(20) |           |          |
Indexes:
"role_pkey" PRIMARY KEY, btree (role_id)
Referenced by:
TABLE "staff" CONSTRAINT "staff_role_fkey" FOREIGN KEY (role) REFERENCES role(role_id)



\d staff

       Table "public.staff"
Column    |          Type          | Collation | Nullable |                Default

-------------+------------------------+-----------+----------+---------------------------------------
    ↪
staff_id    | integer                |           | not null | nextval('staff_staff_id_seq
    ↪ '::regclass)
staff_fname | character varying(25)  |           | not null |
```

```
staff_lname | character varying(35)  |           | not null |
addr1       | character varying(50)  |           | not null |
addr2       | character varying(50)  |           |          |
town        | character varying(60)  |           | not null |
postcode    | character(9)           |           | not null |
home_email  | character varying(255) |           | not null |
work_email  | character varying(100) |           | not null |
role        | integer                |           | not null |
Indexes:
"staff_pkey" PRIMARY KEY, btree (staff_id)
Foreign-key constraints:
"staff_role_fkey" FOREIGN KEY (role) REFERENCES role(role_id)
Referenced by:
TABLE "cust_order" CONSTRAINT "cust_order_staff_id_fkey" FOREIGN KEY (staff_id) REFERENCES
    ↪  staff(staff_id)#
```

12. Compare the printouts to the ERD found on Moodle.
13. Use the ERD to see which tables are related to which table.
14. How many rows of data do you get from the following:

LANGUAGE: SQL

```sql
Select * from product, category;
```

LANGUAGE: Unknown

```
 prod_id |                    prod_name                    | prod_cat | cat_id |
    ↪ cat_name
---------+-------------------------------------------------+----------+--------+-------------
    ↪
       1 | Multi-layered multi-tasking initiative          |        2 |      1 | Men's
    ↪ Wear
       2 | Operative analyzing task-force                  |        1 |      1 | Men's
    ↪ Wear
       3 | Exclusive client-server array                   |        5 |      1 | Men's
    ↪ Wear
       4 | Balanced client-server product                  |        6 |      1 | Men's
    ↪ Wear
       5 | Exclusive background website                    |        5 |      1 | Men's
    ↪ Wear
       6 | Pre-emptive holistic intranet                   |        6 |      1 | Men's
    ↪ Wear
       7 | Re-engineered cohesive methodology              |        1 |      1 | Men's
    ↪ Wear
       8 | Robust directional projection                   |        2 |      1 | Men's
    ↪ Wear
       9 | Inverse transitional infrastructure             |        4 |      1 | Men's
    ↪ Wear
      10 | Multi-tiered explicit paradigm                  |        6 |      1 | Men's
    ↪ Wear
      11 | Re-engineered explicit software                 |        2 |      1 | Men's
    ↪ Wear
      12 | Cross-platform fresh-thinking core              |        3 |      1 | Men's
    ↪ Wear
      13 | Diverse neutral emulation                       |        4 |      1 | Men's
    ↪ Wear
      14 | Up-sized composite challenge                    |        4 |      1 | Men's
    ↪ Wear
      15 | Intuitive directional complexity                |        4 |      1 | Men's
    ↪ Wear
      16 | Universal encompassing conglomeration           |        5 |      1 | Men's
    ↪ Wear
      17 | Multi-channelled well-modulated analyzer        |        2 |      1 | Men's
    ↪ Wear
      18 | Re-engineered actuating capability              |        4 |      1 | Men's
    ↪ Wear
      19 | Public-key interactive encoding                 |        2 |      1 | Men's
```

```
    ↪ Wear
23    20 | Monitored asynchronous function                    |      6 |      1 | Men's
    ↪ Wear
24    21 | Proactive methodical data-warehouse               |      4 |      1 | Men's
    ↪ Wear
25    22 | Balanced real-time info-mediaries                 |      1 |      1 | Men's
    ↪ Wear
26    23 | Right-sized mission-critical pricing structure    |      6 |      1 | Men's
    ↪ Wear
27    24 | Synergistic homogeneous ability                   |      5 |      1 | Men's
    ↪ Wear
28    25 | Open-source impactful archive                     |      5 |      1 | Men's
    ↪ Wear
29    26 | Realigned 5th generation artificial intelligence  |      2 |      1 | Men's
    ↪ Wear
30    27 | Configurable methodical firmware                  |      5 |      1 | Men's
    ↪ Wear
31    28 | Profound optimal encryption                       |      3 |      1 | Men's
    ↪ Wear
32    29 | Vision-oriented user-facing framework             |      2 |      1 | Men's
    ↪ Wear
33    30 | Secured holistic hierarchy                        |      2 |      1 | Men's
    ↪ Wear
34    31 | Assimilated regional instruction set              |      2 |      1 | Men's
    ↪ Wear
35    32 | Business-focused holistic help-desk               |      3 |      1 | Men's
    ↪ Wear
36    33 | Virtual stable Graphic Interface                  |      5 |      1 | Men's
    ↪ Wear
37    34 | Implemented optimizing benchmark                  |      1 |      1 | Men's
    ↪ Wear
38    35 | Adaptive static website                           |      1 |      1 | Men's
    ↪ Wear
39    36 | Virtual impactful success                         |      2 |      1 | Men's
    ↪ Wear
40    37 | Open-architected homogeneous concept              |      6 |      1 | Men's
    ↪ Wear
41    38 | Diverse reciprocal knowledge base                 |      1 |      1 | Men's
    ↪ Wear
42    39 | Realigned homogeneous hub                         |      5 |      1 | Men's
    ↪ Wear
43    40 | Switchable tangible product                       |      4 |      1 | Men's
    ↪ Wear
44    41 | Universal global hub                              |      2 |      1 | Men's
    ↪ Wear
45    42 | Enhanced discrete function                        |      4 |      1 | Men's
    ↪ Wear
46    43 | Horizontal asynchronous intranet                  |      4 |      1 | Men's
    ↪ Wear
47    44 | Quality-focused foreground analyzer               |      5 |      1 | Men's
    ↪ Wear
48    45 | Configurable analyzing solution                   |      3 |      1 | Men's
    ↪ Wear
49    46 | Fully-configurable full-range interface           |      6 |      1 | Men's
    ↪ Wear
50    47 | Monitored non-volatile initiative                 |      3 |      1 | Men's
    ↪ Wear
51    48 | Pre-emptive next generation infrastructure        |      3 |      1 | Men's
    ↪ Wear
52    49 | Switchable 5th generation parallelism             |      4 |      1 | Men's
    ↪ Wear
53    50 | Adaptive modular approach                         |      2 |      1 | Men's
    ↪ Wear
54    51 | Synergistic zero defect info-mediaries            |      2 |      1 | Men's
    ↪ Wear
55    52 | Persevering empowering customer loyalty           |      3 |      1 | Men's
    ↪ Wear
56    53 | Robust foreground leverage                        |      1 |      1 | Men's
    ↪ Wear
57    54 | Customizable cohesive capacity                    |      6 |      1 | Men's
    ↪ Wear
58    55 | Progressive modular archive                       |      3 |      1 | Men's
    ↪ Wear
59    56 | Reduced fresh-thinking process improvement        |      2 |      1 | Men's
    ↪ Wear
```

```
50      57 | Seamless optimal leverage                          |       6 |       1 | Men's
    ↪ Wear
51      58 | Universal exuding protocol                         |       5 |       1 | Men's
    ↪ Wear
52      59 | Realigned client-driven database                   |       6 |       1 | Men's
    ↪ Wear
53      60 | Balanced hybrid portal                             |       5 |       1 | Men's
    ↪ Wear
54      61 | Customizable well-modulated encryption             |       5 |       1 | Men's
    ↪ Wear
55      62 | Cross-group reciprocal firmware                    |       3 |       1 | Men's
    ↪ Wear
56      63 | 4th generation Graphical User Interface            |       4 |       1 | Men's
    ↪ Wear
57      64 | Business-focused solution-oriented moratorium      |       5 |       1 | Men's
    ↪ Wear
58      65 | Synergistic scalable capability                    |       5 |       1 | Men's
    ↪ Wear
59      66 | Distributed uniform Graphic Interface              |       5 |       1 | Men's
    ↪ Wear
60      67 | Stand-alone composite Graphical User Interface     |       2 |       1 | Men's
    ↪ Wear
61      68 | Future-proofed leading edge customer loyalty       |       4 |       1 | Men's
    ↪ Wear
62      69 | Profound human-resource forecast                   |       6 |       1 | Men's
    ↪ Wear
63      70 | Advanced neutral portal                            |       3 |       1 | Men's
    ↪ Wear
64      71 | Customer-focused needs-based protocol              |       3 |       1 | Men's
    ↪ Wear
65      72 | User-friendly encompassing array                   |       6 |       1 | Men's
    ↪ Wear
66      73 | Decentralized human-resource infrastructure        |       2 |       1 | Men's
    ↪ Wear
67      74 | Balanced modular website                           |       2 |       1 | Men's
    ↪ Wear
68      75 | Horizontal explicit benchmark                      |       2 |       1 | Men's
    ↪ Wear
69      76 | Re-engineered 24/7 knowledge base                  |       1 |       1 | Men's
    ↪ Wear
70      77 | Innovative web-enabled extranet                    |       2 |       1 | Men's
    ↪ Wear
71      78 | Exclusive analyzing open architecture              |       2 |       1 | Men's
    ↪ Wear
72      79 | Fundamental global archive                         |       3 |       1 | Men's
    ↪ Wear
73      80 | Profound value-added intranet                      |       5 |       1 | Men's
    ↪ Wear
74      81 | Networked global open system                       |       6 |       1 | Men's
    ↪ Wear
75      82 | Persistent demand-driven complexity                |       5 |       1 | Men's
    ↪ Wear
76      83 | Focused secondary initiative                       |       5 |       1 | Men's
    ↪ Wear
77      84 | Digitized tertiary groupware                       |       3 |       1 | Men's
    ↪ Wear
78      85 | Enhanced homogeneous paradigm                      |       4 |       1 | Men's
    ↪ Wear
79      86 | Inverse high-level attitude                        |       4 |       1 | Men's
    ↪ Wear
80      87 | Quality-focused upward-trending throughput         |       4 |       1 | Men's
    ↪ Wear
81      88 | Team-oriented stable project                       |       6 |       1 | Men's
    ↪ Wear
82      89 | Total intangible artificial intelligence           |       3 |       1 | Men's
    ↪ Wear
83      90 | Streamlined asynchronous functionalities           |       5 |       1 | Men's
    ↪ Wear
84      91 | Vision-oriented attitude-oriented core             |       5 |       1 | Men's
    ↪ Wear
85      92 | Integrated 24/7 interface                          |       2 |       1 | Men's
    ↪ Wear
86      93 | Advanced didactic Graphic Interface                |       1 |       1 | Men's
    ↪ Wear
87      94 | Exclusive multimedia middleware                    |       6 |       1 | Men's
```

```
    ↪ Wear
98    95 | Ameliorated next generation orchestration        |      6 |     1 | Men's
    ↪ Wear
99    96 | Front-line demand-driven utilisation            |      5 |     1 | Men's
    ↪ Wear
100   97 | Organic clear-thinking system engine            |      3 |     1 | Men's
    ↪ Wear
101   98 | Persistent incremental model                    |      3 |     1 | Men's
    ↪ Wear
102   99 | Ergonomic solution-oriented local area network  |      2 |     1 | Men's
    ↪ Wear
103  100 | Robust mission-critical complexity              |      5 |     1 | Men's
    ↪ Wear
104    1 | Multi-layered multi-tasking initiative          |      2 |     2 | Ladies
    ↪ Wear
105    2 | Operative analyzing task-force                  |      1 |     2 | Ladies
    ↪ Wear
106    3 | Exclusive client-server array                   |      5 |     2 | Ladies
    ↪ Wear
107    4 | Balanced client-server product                  |      6 |     2 | Ladies
    ↪ Wear
108    5 | Exclusive background website                    |      5 |     2 | Ladies
    ↪ Wear
109    6 | Pre-emptive holistic intranet                   |      6 |     2 | Ladies
    ↪ Wear
110    7 | Re-engineered cohesive methodology              |      1 |     2 | Ladies
    ↪ Wear
111    8 | Robust directional projection                   |      2 |     2 | Ladies
    ↪ Wear
112    9 | Inverse transitional infrastructure             |      4 |     2 | Ladies
    ↪ Wear
113   10 | Multi-tiered explicit paradigm                  |      6 |     2 | Ladies
    ↪ Wear
114   11 | Re-engineered explicit software                 |      2 |     2 | Ladies
    ↪ Wear
115   12 | Cross-platform fresh-thinking core              |      3 |     2 | Ladies
    ↪ Wear
116   13 | Diverse neutral emulation                       |      4 |     2 | Ladies
    ↪ Wear
117   14 | Up-sized composite challenge                    |      4 |     2 | Ladies
    ↪ Wear
118   15 | Intuitive directional complexity                |      4 |     2 | Ladies
    ↪ Wear
119   16 | Universal encompassing conglomeration           |      5 |     2 | Ladies
    ↪ Wear
120   17 | Multi-channelled well-modulated analyzer        |      2 |     2 | Ladies
    ↪ Wear
121   18 | Re-engineered actuating capability              |      4 |     2 | Ladies
    ↪ Wear
122   19 | Public-key interactive encoding                 |      2 |     2 | Ladies
    ↪ Wear
123   20 | Monitored asynchronous function                 |      6 |     2 | Ladies
    ↪ Wear
124   21 | Proactive methodical data-warehouse             |      4 |     2 | Ladies
    ↪ Wear
125   22 | Balanced real-time info-mediaries               |      1 |     2 | Ladies
    ↪ Wear
126   23 | Right-sized mission-critical pricing structure  |      6 |     2 | Ladies
    ↪ Wear
127   24 | Synergistic homogeneous ability                 |      5 |     2 | Ladies
    ↪ Wear
128   25 | Open-source impactful archive                   |      5 |     2 | Ladies
    ↪ Wear
129   26 | Realigned 5th generation artificial intelligence |      2 |     2 | Ladies
    ↪ Wear
130   27 | Configurable methodical firmware                |      5 |     2 | Ladies
    ↪ Wear
131   28 | Profound optimal encryption                     |      3 |     2 | Ladies
    ↪ Wear
132   29 | Vision-oriented user-facing framework           |      2 |     2 | Ladies
    ↪ Wear
133   30 | Secured holistic hierarchy                      |      2 |     2 | Ladies
    ↪ Wear
134   31 | Assimilated regional instruction set            |      2 |     2 | Ladies
    ↪ Wear
```

```
      32 | Business-focused holistic help-desk                |        3 |       2 | Ladies
    ↪ Wear
      33 | Virtual stable Graphic Interface                   |        5 |       2 | Ladies
    ↪ Wear
      34 | Implemented optimizing benchmark                   |        1 |       2 | Ladies
    ↪ Wear
      35 | Adaptive static website                            |        1 |       2 | Ladies
    ↪ Wear
      36 | Virtual impactful success                          |        2 |       2 | Ladies
    ↪ Wear
      37 | Open-architected homogeneous concept               |        6 |       2 | Ladies
    ↪ Wear
      38 | Diverse reciprocal knowledge base                  |        1 |       2 | Ladies
    ↪ Wear
      39 | Realigned homogeneous hub                          |        5 |       2 | Ladies
    ↪ Wear
      40 | Switchable tangible product                        |        4 |       2 | Ladies
    ↪ Wear
      41 | Universal global hub                               |        2 |       2 | Ladies
    ↪ Wear
      42 | Enhanced discrete function                         |        4 |       2 | Ladies
    ↪ Wear
      43 | Horizontal asynchronous intranet                   |        4 |       2 | Ladies
    ↪ Wear
      44 | Quality-focused foreground analyzer                |        5 |       2 | Ladies
    ↪ Wear
      45 | Configurable analyzing solution                    |        3 |       2 | Ladies
    ↪ Wear
      46 | Fully-configurable full-range interface            |        6 |       2 | Ladies
    ↪ Wear
      47 | Monitored non-volatile initiative                  |        3 |       2 | Ladies
    ↪ Wear
      48 | Pre-emptive next generation infrastructure         |        3 |       2 | Ladies
    ↪ Wear
      49 | Switchable 5th generation parallelism              |        4 |       2 | Ladies
    ↪ Wear
      50 | Adaptive modular approach                          |        2 |       2 | Ladies
    ↪ Wear
      51 | Synergistic zero defect info-mediaries             |        2 |       2 | Ladies
    ↪ Wear
      52 | Persevering empowering customer loyalty            |        3 |       2 | Ladies
    ↪ Wear
      53 | Robust foreground leverage                         |        1 |       2 | Ladies
    ↪ Wear
      54 | Customizable cohesive capacity                     |        6 |       2 | Ladies
    ↪ Wear
      55 | Progressive modular archive                        |        3 |       2 | Ladies
    ↪ Wear
      56 | Reduced fresh-thinking process improvement         |        2 |       2 | Ladies
    ↪ Wear
      57 | Seamless optimal leverage                          |        6 |       2 | Ladies
    ↪ Wear
      58 | Universal exuding protocol                         |        5 |       2 | Ladies
    ↪ Wear
      59 | Realigned client-driven database                   |        6 |       2 | Ladies
    ↪ Wear
      60 | Balanced hybrid portal                             |        5 |       2 | Ladies
    ↪ Wear
      61 | Customizable well-modulated encryption             |        5 |       2 | Ladies
    ↪ Wear
      62 | Cross-group reciprocal firmware                    |        3 |       2 | Ladies
    ↪ Wear
      63 | 4th generation Graphical User Interface            |        4 |       2 | Ladies
    ↪ Wear
      64 | Business-focused solution-oriented moratorium      |        5 |       2 | Ladies
    ↪ Wear
      65 | Synergistic scalable capability                    |        5 |       2 | Ladies
    ↪ Wear
      66 | Distributed uniform Graphic Interface              |        5 |       2 | Ladies
    ↪ Wear
      67 | Stand-alone composite Graphical User Interface     |        2 |       2 | Ladies
    ↪ Wear
      68 | Future-proofed leading edge customer loyalty       |        4 |       2 | Ladies
    ↪ Wear
      69 | Profound human-resource forecast                   |        6 |       2 | Ladies
```

```
    ↪ Wear
173   70 | Advanced neutral portal                          |       3 |      2 | Ladies
    ↪ Wear
174   71 | Customer-focused needs-based protocol           |       3 |      2 | Ladies
    ↪ Wear
175   72 | User-friendly encompassing array                |       6 |      2 | Ladies
    ↪ Wear
176   73 | Decentralized human-resource infrastructure     |       2 |      2 | Ladies
    ↪ Wear
177   74 | Balanced modular website                        |       2 |      2 | Ladies
    ↪ Wear
178   75 | Horizontal explicit benchmark                   |       2 |      2 | Ladies
    ↪ Wear
179   76 | Re-engineered 24/7 knowledge base               |       1 |      2 | Ladies
    ↪ Wear
180   77 | Innovative web-enabled extranet                 |       2 |      2 | Ladies
    ↪ Wear
181   78 | Exclusive analyzing open architecture           |       2 |      2 | Ladies
    ↪ Wear
182   79 | Fundamental global archive                      |       3 |      2 | Ladies
    ↪ Wear
183   80 | Profound value-added intranet                   |       5 |      2 | Ladies
    ↪ Wear
184   81 | Networked global open system                    |       6 |      2 | Ladies
    ↪ Wear
185   82 | Persistent demand-driven complexity             |       5 |      2 | Ladies
    ↪ Wear
186   83 | Focused secondary initiative                    |       5 |      2 | Ladies
    ↪ Wear
187   84 | Digitized tertiary groupware                    |       3 |      2 | Ladies
    ↪ Wear
188   85 | Enhanced homogeneous paradigm                   |       4 |      2 | Ladies
    ↪ Wear
189   86 | Inverse high-level attitude                     |       4 |      2 | Ladies
    ↪ Wear
190   87 | Quality-focused upward-trending throughput      |       4 |      2 | Ladies
    ↪ Wear
191   88 | Team-oriented stable project                    |       6 |      2 | Ladies
    ↪ Wear
192   89 | Total intangible artificial intelligence        |       3 |      2 | Ladies
    ↪ Wear
193   90 | Streamlined asynchronous functionalities        |       5 |      2 | Ladies
    ↪ Wear
194   91 | Vision-oriented attitude-oriented core          |       5 |      2 | Ladies
    ↪ Wear
195   92 | Integrated 24/7 interface                       |       2 |      2 | Ladies
    ↪ Wear
196   93 | Advanced didactic Graphic Interface             |       1 |      2 | Ladies
    ↪ Wear
197   94 | Exclusive multimedia middleware                 |       6 |      2 | Ladies
    ↪ Wear
198   95 | Ameliorated next generation orchestration       |       6 |      2 | Ladies
    ↪ Wear
199   96 | Front-line demand-driven utilisation            |       5 |      2 | Ladies
    ↪ Wear
200   97 | Organic clear-thinking system engine            |       3 |      2 | Ladies
    ↪ Wear
201   98 | Persistent incremental model                    |       3 |      2 | Ladies
    ↪ Wear
202   99 | Ergonomic solution-oriented local area network  |       2 |      2 | Ladies
    ↪ Wear
203  100 | Robust mission-critical complexity              |       5 |      2 | Ladies
    ↪ Wear
204    1 | Multi-layered multi-tasking initiative          |       2 |      3 | Kid's
    ↪ Wear
205    2 | Operative analyzing task-force                  |       1 |      3 | Kid's
    ↪ Wear
206    3 | Exclusive client-server array                   |       5 |      3 | Kid's
    ↪ Wear
207    4 | Balanced client-server product                  |       6 |      3 | Kid's
    ↪ Wear
208    5 | Exclusive background website                    |       5 |      3 | Kid's
    ↪ Wear
209    6 | Pre-emptive holistic intranet                   |       6 |      3 | Kid's
    ↪ Wear
```

```
      7 | Re-engineered cohesive methodology                |       1 |       3 | Kid's
    ↪ Wear
      8 | Robust directional projection                     |       2 |       3 | Kid's
    ↪ Wear
      9 | Inverse transitional infrastructure               |       4 |       3 | Kid's
    ↪ Wear
     10 | Multi-tiered explicit paradigm                    |       6 |       3 | Kid's
    ↪ Wear
     11 | Re-engineered explicit software                   |       2 |       3 | Kid's
    ↪ Wear
     12 | Cross-platform fresh-thinking core                |       3 |       3 | Kid's
    ↪ Wear
     13 | Diverse neutral emulation                         |       4 |       3 | Kid's
    ↪ Wear
     14 | Up-sized composite challenge                      |       4 |       3 | Kid's
    ↪ Wear
     15 | Intuitive directional complexity                  |       4 |       3 | Kid's
    ↪ Wear
     16 | Universal encompassing conglomeration             |       5 |       3 | Kid's
    ↪ Wear
     17 | Multi-channelled well-modulated analyzer          |       2 |       3 | Kid's
    ↪ Wear
     18 | Re-engineered actuating capability                |       4 |       3 | Kid's
    ↪ Wear
     19 | Public-key interactive encoding                   |       2 |       3 | Kid's
    ↪ Wear
     20 | Monitored asynchronous function                   |       6 |       3 | Kid's
    ↪ Wear
     21 | Proactive methodical data-warehouse               |       4 |       3 | Kid's
    ↪ Wear
     22 | Balanced real-time info-mediaries                 |       1 |       3 | Kid's
    ↪ Wear
     23 | Right-sized mission-critical pricing structure    |       6 |       3 | Kid's
    ↪ Wear
     24 | Synergistic homogeneous ability                   |       5 |       3 | Kid's
    ↪ Wear
     25 | Open-source impactful archive                     |       5 |       3 | Kid's
    ↪ Wear
     26 | Realigned 5th generation artificial intelligence  |       2 |       3 | Kid's
    ↪ Wear
     27 | Configurable methodical firmware                  |       5 |       3 | Kid's
    ↪ Wear
     28 | Profound optimal encryption                       |       3 |       3 | Kid's
    ↪ Wear
     29 | Vision-oriented user-facing framework             |       2 |       3 | Kid's
    ↪ Wear
     30 | Secured holistic hierarchy                        |       2 |       3 | Kid's
    ↪ Wear
     31 | Assimilated regional instruction set              |       2 |       3 | Kid's
    ↪ Wear
     32 | Business-focused holistic help-desk               |       3 |       3 | Kid's
    ↪ Wear
     33 | Virtual stable Graphic Interface                  |       5 |       3 | Kid's
    ↪ Wear
     34 | Implemented optimizing benchmark                  |       1 |       3 | Kid's
    ↪ Wear
     35 | Adaptive static website                           |       1 |       3 | Kid's
    ↪ Wear
     36 | Virtual impactful success                         |       2 |       3 | Kid's
    ↪ Wear
     37 | Open-architected homogeneous concept              |       6 |       3 | Kid's
    ↪ Wear
     38 | Diverse reciprocal knowledge base                 |       1 |       3 | Kid's
    ↪ Wear
     39 | Realigned homogeneous hub                         |       5 |       3 | Kid's
    ↪ Wear
     40 | Switchable tangible product                       |       4 |       3 | Kid's
    ↪ Wear
     41 | Universal global hub                              |       2 |       3 | Kid's
    ↪ Wear
     42 | Enhanced discrete function                        |       4 |       3 | Kid's
    ↪ Wear
     43 | Horizontal asynchronous intranet                  |       4 |       3 | Kid's
    ↪ Wear
     44 | Quality-focused foreground analyzer               |       5 |       3 | Kid's
```

```
    ↪ Wear
218   45 | Configurable analyzing solution                |        3 |      3 | Kid's
    ↪ Wear
219   46 | Fully-configurable full-range interface        |        6 |      3 | Kid's
    ↪ Wear
220   47 | Monitored non-volatile initiative              |        3 |      3 | Kid's
    ↪ Wear
221   48 | Pre-emptive next generation infrastructure     |        3 |      3 | Kid's
    ↪ Wear
222   49 | Switchable 5th generation parallelism          |        4 |      3 | Kid's
    ↪ Wear
223   50 | Adaptive modular approach                      |        2 |      3 | Kid's
    ↪ Wear
224   51 | Synergistic zero defect info-mediaries         |        2 |      3 | Kid's
    ↪ Wear
225   52 | Persevering empowering customer loyalty        |        3 |      3 | Kid's
    ↪ Wear
226   53 | Robust foreground leverage                     |        1 |      3 | Kid's
    ↪ Wear
227   54 | Customizable cohesive capacity                 |        6 |      3 | Kid's
    ↪ Wear
228   55 | Progressive modular archive                    |        3 |      3 | Kid's
    ↪ Wear
229   56 | Reduced fresh-thinking process improvement     |        2 |      3 | Kid's
    ↪ Wear
230   57 | Seamless optimal leverage                      |        6 |      3 | Kid's
    ↪ Wear
231   58 | Universal exuding protocol                     |        5 |      3 | Kid's
    ↪ Wear
232   59 | Realigned client-driven database               |        6 |      3 | Kid's
    ↪ Wear
233   60 | Balanced hybrid portal                         |        5 |      3 | Kid's
    ↪ Wear
234   61 | Customizable well-modulated encryption         |        5 |      3 | Kid's
    ↪ Wear
235   62 | Cross-group reciprocal firmware                |        3 |      3 | Kid's
    ↪ Wear
236   63 | 4th generation Graphical User Interface        |        4 |      3 | Kid's
    ↪ Wear
237   64 | Business-focused solution-oriented moratorium  |        5 |      3 | Kid's
    ↪ Wear
238   65 | Synergistic scalable capability                |        5 |      3 | Kid's
    ↪ Wear
239   66 | Distributed uniform Graphic Interface          |        5 |      3 | Kid's
    ↪ Wear
240   67 | Stand-alone composite Graphical User Interface |        2 |      3 | Kid's
    ↪ Wear
241   68 | Future-proofed leading edge customer loyalty   |        4 |      3 | Kid's
    ↪ Wear
242   69 | Profound human-resource forecast               |        6 |      3 | Kid's
    ↪ Wear
243   70 | Advanced neutral portal                        |        3 |      3 | Kid's
    ↪ Wear
244   71 | Customer-focused needs-based protocol          |        3 |      3 | Kid's
    ↪ Wear
245   72 | User-friendly encompassing array               |        6 |      3 | Kid's
    ↪ Wear
246   73 | Decentralized human-resource infrastructure    |        2 |      3 | Kid's
    ↪ Wear
247   74 | Balanced modular website                       |        2 |      3 | Kid's
    ↪ Wear
248   75 | Horizontal explicit benchmark                  |        2 |      3 | Kid's
    ↪ Wear
249   76 | Re-engineered 24/7 knowledge base              |        1 |      3 | Kid's
    ↪ Wear
250   77 | Innovative web-enabled extranet                |        2 |      3 | Kid's
    ↪ Wear
251   78 | Exclusive analyzing open architecture          |        2 |      3 | Kid's
    ↪ Wear
252   79 | Fundamental global archive                     |        3 |      3 | Kid's
    ↪ Wear
253   80 | Profound value-added intranet                  |        5 |      3 | Kid's
    ↪ Wear
254   81 | Networked global open system                   |        6 |      3 | Kid's
    ↪ Wear
```

```
285       82 | Persistent demand-driven complexity              |         5 |       3 | Kid's
      ↪ Wear
286       83 | Focused secondary initiative                     |         5 |       3 | Kid's
      ↪ Wear
287       84 | Digitized tertiary groupware                     |         3 |       3 | Kid's
      ↪ Wear
288       85 | Enhanced homogeneous paradigm                    |         4 |       3 | Kid's
      ↪ Wear
289       86 | Inverse high-level attitude                      |         4 |       3 | Kid's
      ↪ Wear
290       87 | Quality-focused upward-trending throughput       |         4 |       3 | Kid's
      ↪ Wear
291       88 | Team-oriented stable project                     |         6 |       3 | Kid's
      ↪ Wear
292       89 | Total intangible artificial intelligence         |         3 |       3 | Kid's
      ↪ Wear
293       90 | Streamlined asynchronous functionalities         |         5 |       3 | Kid's
      ↪ Wear
294       91 | Vision-oriented attitude-oriented core           |         5 |       3 | Kid's
      ↪ Wear
295       92 | Integrated 24/7 interface                        |         2 |       3 | Kid's
      ↪ Wear
296       93 | Advanced didactic Graphic Interface              |         1 |       3 | Kid's
      ↪ Wear
297       94 | Exclusive multimedia middleware                  |         6 |       3 | Kid's
      ↪ Wear
298       95 | Ameliorated next generation orchestration        |         6 |       3 | Kid's
      ↪ Wear
299       96 | Front-line demand-driven utilisation             |         5 |       3 | Kid's
      ↪ Wear
300       97 | Organic clear-thinking system engine             |         3 |       3 | Kid's
      ↪ Wear
301       98 | Persistent incremental model                     |         3 |       3 | Kid's
      ↪ Wear
302       99 | Ergonomic solution-oriented local area network   |         2 |       3 | Kid's
      ↪ Wear
303      100 | Robust mission-critical complexity               |         5 |       3 | Kid's
      ↪ Wear
304        1 | Multi-layered multi-tasking initiative           |         2 |       4 | Outdoor
305        2 | Operative analyzing task-force                   |         1 |       4 | Outdoor
306        3 | Exclusive client-server array                    |         5 |       4 | Outdoor
307        4 | Balanced client-server product                   |         6 |       4 | Outdoor
308        5 | Exclusive background website                     |         5 |       4 | Outdoor
309        6 | Pre-emptive holistic intranet                    |         6 |       4 | Outdoor
310        7 | Re-engineered cohesive methodology               |         1 |       4 | Outdoor
311        8 | Robust directional projection                    |         2 |       4 | Outdoor
312        9 | Inverse transitional infrastructure              |         4 |       4 | Outdoor
313       10 | Multi-tiered explicit paradigm                   |         6 |       4 | Outdoor
314       11 | Re-engineered explicit software                  |         2 |       4 | Outdoor
315       12 | Cross-platform fresh-thinking core               |         3 |       4 | Outdoor
316       13 | Diverse neutral emulation                        |         4 |       4 | Outdoor
317       14 | Up-sized composite challenge                     |         4 |       4 | Outdoor
318       15 | Intuitive directional complexity                 |         4 |       4 | Outdoor
319       16 | Universal encompassing conglomeration            |         5 |       4 | Outdoor
320       17 | Multi-channelled well-modulated analyzer         |         2 |       4 | Outdoor
321       18 | Re-engineered actuating capability               |         4 |       4 | Outdoor
322       19 | Public-key interactive encoding                  |         2 |       4 | Outdoor
323       20 | Monitored asynchronous function                  |         6 |       4 | Outdoor
324       21 | Proactive methodical data-warehouse              |         4 |       4 | Outdoor
325       22 | Balanced real-time info-mediaries                |         1 |       4 | Outdoor
326       23 | Right-sized mission-critical pricing structure   |         6 |       4 | Outdoor
327       24 | Synergistic homogeneous ability                  |         5 |       4 | Outdoor
328       25 | Open-source impactful archive                    |         5 |       4 | Outdoor
329       26 | Realigned 5th generation artificial intelligence |         2 |       4 | Outdoor
330       27 | Configurable methodical firmware                 |         5 |       4 | Outdoor
331       28 | Profound optimal encryption                      |         3 |       4 | Outdoor
332       29 | Vision-oriented user-facing framework            |         2 |       4 | Outdoor
333       30 | Secured holistic hierarchy                       |         2 |       4 | Outdoor
334       31 | Assimilated regional instruction set             |         2 |       4 | Outdoor
335       32 | Business-focused holistic help-desk              |         3 |       4 | Outdoor
336       33 | Virtual stable Graphic Interface                 |         5 |       4 | Outdoor
337       34 | Implemented optimizing benchmark                 |         1 |       4 | Outdoor
338       35 | Adaptive static website                          |         1 |       4 | Outdoor
339       36 | Virtual impactful success                        |         2 |       4 | Outdoor
340       37 | Open-architected homogeneous concept             |         6 |       4 | Outdoor
```

```
341      38 | Diverse reciprocal knowledge base                |        1 |       4 | Outdoor
342      39 | Realigned homogeneous hub                        |        5 |       4 | Outdoor
343      40 | Switchable tangible product                      |        4 |       4 | Outdoor
344      41 | Universal global hub                             |        2 |       4 | Outdoor
345      42 | Enhanced discrete function                       |        4 |       4 | Outdoor
346      43 | Horizontal asynchronous intranet                 |        4 |       4 | Outdoor
347      44 | Quality-focused foreground analyzer              |        5 |       4 | Outdoor
348      45 | Configurable analyzing solution                  |        3 |       4 | Outdoor
349      46 | Fully-configurable full-range interface          |        6 |       4 | Outdoor
350      47 | Monitored non-volatile initiative                |        3 |       4 | Outdoor
351      48 | Pre-emptive next generation infrastructure       |        3 |       4 | Outdoor
352      49 | Switchable 5th generation parallelism            |        4 |       4 | Outdoor
353      50 | Adaptive modular approach                        |        2 |       4 | Outdoor
354      51 | Synergistic zero defect info-mediaries           |        2 |       4 | Outdoor
355      52 | Persevering empowering customer loyalty          |        3 |       4 | Outdoor
356      53 | Robust foreground leverage                       |        1 |       4 | Outdoor
357      54 | Customizable cohesive capacity                   |        6 |       4 | Outdoor
358      55 | Progressive modular archive                      |        3 |       4 | Outdoor
359      56 | Reduced fresh-thinking process improvement       |        2 |       4 | Outdoor
360      57 | Seamless optimal leverage                        |        6 |       4 | Outdoor
361      58 | Universal exuding protocol                       |        5 |       4 | Outdoor
362      59 | Realigned client-driven database                 |        6 |       4 | Outdoor
363      60 | Balanced hybrid portal                           |        5 |       4 | Outdoor
364      61 | Customizable well-modulated encryption           |        5 |       4 | Outdoor
365      62 | Cross-group reciprocal firmware                  |        3 |       4 | Outdoor
366      63 | 4th generation Graphical User Interface          |        4 |       4 | Outdoor
367      64 | Business-focused solution-oriented moratorium    |        5 |       4 | Outdoor
368      65 | Synergistic scalable capability                  |        5 |       4 | Outdoor
369      66 | Distributed uniform Graphic Interface            |        5 |       4 | Outdoor
370      67 | Stand-alone composite Graphical User Interface   |        2 |       4 | Outdoor
371      68 | Future-proofed leading edge customer loyalty     |        4 |       4 | Outdoor
372      69 | Profound human-resource forecast                 |        6 |       4 | Outdoor
373      70 | Advanced neutral portal                          |        3 |       4 | Outdoor
374      71 | Customer-focused needs-based protocol            |        3 |       4 | Outdoor
375      72 | User-friendly encompassing array                 |        6 |       4 | Outdoor
376      73 | Decentralized human-resource infrastructure      |        2 |       4 | Outdoor
377      74 | Balanced modular website                         |        2 |       4 | Outdoor
378      75 | Horizontal explicit benchmark                    |        2 |       4 | Outdoor
379      76 | Re-engineered 24/7 knowledge base                |        1 |       4 | Outdoor
380      77 | Innovative web-enabled extranet                  |        2 |       4 | Outdoor
381      78 | Exclusive analyzing open architecture            |        2 |       4 | Outdoor
382      79 | Fundamental global archive                       |        3 |       4 | Outdoor
383      80 | Profound value-added intranet                    |        5 |       4 | Outdoor
384      81 | Networked global open system                     |        6 |       4 | Outdoor
385      82 | Persistent demand-driven complexity              |        5 |       4 | Outdoor
386      83 | Focused secondary initiative                     |        5 |       4 | Outdoor
387      84 | Digitized tertiary groupware                     |        3 |       4 | Outdoor
388      85 | Enhanced homogeneous paradigm                    |        4 |       4 | Outdoor
389      86 | Inverse high-level attitude                      |        4 |       4 | Outdoor
390      87 | Quality-focused upward-trending throughput       |        4 |       4 | Outdoor
391      88 | Team-oriented stable project                     |        6 |       4 | Outdoor
392      89 | Total intangible artificial intelligence         |        3 |       4 | Outdoor
393      90 | Streamlined asynchronous functionalities         |        5 |       4 | Outdoor
394      91 | Vision-oriented attitude-oriented core           |        5 |       4 | Outdoor
395      92 | Integrated 24/7 interface                        |        2 |       4 | Outdoor
396      93 | Advanced didactic Graphic Interface              |        1 |       4 | Outdoor
397      94 | Exclusive multimedia middleware                  |        6 |       4 | Outdoor
398      95 | Ameliorated next generation orchestration        |        6 |       4 | Outdoor
399      96 | Front-line demand-driven utilisation             |        5 |       4 | Outdoor
400      97 | Organic clear-thinking system engine             |        3 |       4 | Outdoor
401      98 | Persistent incremental model                     |        3 |       4 | Outdoor
402      99 | Ergonomic solution-oriented local area network   |        2 |       4 | Outdoor
403     100 | Robust mission-critical complexity               |        5 |       4 | Outdoor
404       1 | Multi-layered multi-tasking initiative           |        2 |       5 | Sport
405       2 | Operative analyzing task-force                   |        1 |       5 | Sport
406       3 | Exclusive client-server array                    |        5 |       5 | Sport
407       4 | Balanced client-server product                   |        6 |       5 | Sport
408       5 | Exclusive background website                     |        5 |       5 | Sport
409       6 | Pre-emptive holistic intranet                    |        6 |       5 | Sport
410       7 | Re-engineered cohesive methodology               |        1 |       5 | Sport
411       8 | Robust directional projection                    |        2 |       5 | Sport
412       9 | Inverse transitional infrastructure              |        4 |       5 | Sport
413      10 | Multi-tiered explicit paradigm                   |        6 |       5 | Sport
414      11 | Re-engineered explicit software                  |        2 |       5 | Sport
415      12 | Cross-platform fresh-thinking core               |        3 |       5 | Sport
```

```
      13 | Diverse neutral emulation                          |       4 |       5 | Sport
      14 | Up-sized composite challenge                       |       4 |       5 | Sport
      15 | Intuitive directional complexity                   |       4 |       5 | Sport
      16 | Universal encompassing conglomeration              |       5 |       5 | Sport
      17 | Multi-channelled well-modulated analyzer           |       2 |       5 | Sport
      18 | Re-engineered actuating capability                 |       4 |       5 | Sport
      19 | Public-key interactive encoding                    |       2 |       5 | Sport
      20 | Monitored asynchronous function                    |       6 |       5 | Sport
      21 | Proactive methodical data-warehouse                |       4 |       5 | Sport
      22 | Balanced real-time info-mediaries                  |       1 |       5 | Sport
      23 | Right-sized mission-critical pricing structure     |       6 |       5 | Sport
      24 | Synergistic homogeneous ability                    |       5 |       5 | Sport
      25 | Open-source impactful archive                      |       5 |       5 | Sport
      26 | Realigned 5th generation artificial intelligence   |       2 |       5 | Sport
      27 | Configurable methodical firmware                   |       5 |       5 | Sport
      28 | Profound optimal encryption                        |       3 |       5 | Sport
      29 | Vision-oriented user-facing framework              |       2 |       5 | Sport
      30 | Secured holistic hierarchy                         |       2 |       5 | Sport
      31 | Assimilated regional instruction set               |       2 |       5 | Sport
      32 | Business-focused holistic help-desk                |       3 |       5 | Sport
      33 | Virtual stable Graphic Interface                   |       5 |       5 | Sport
      34 | Implemented optimizing benchmark                   |       1 |       5 | Sport
      35 | Adaptive static website                            |       1 |       5 | Sport
      36 | Virtual impactful success                          |       2 |       5 | Sport
      37 | Open-architected homogeneous concept               |       6 |       5 | Sport
      38 | Diverse reciprocal knowledge base                  |       1 |       5 | Sport
      39 | Realigned homogeneous hub                          |       5 |       5 | Sport
      40 | Switchable tangible product                        |       4 |       5 | Sport
      41 | Universal global hub                               |       2 |       5 | Sport
      42 | Enhanced discrete function                         |       4 |       5 | Sport
      43 | Horizontal asynchronous intranet                   |       4 |       5 | Sport
      44 | Quality-focused foreground analyzer                |       5 |       5 | Sport
      45 | Configurable analyzing solution                    |       3 |       5 | Sport
      46 | Fully-configurable full-range interface            |       6 |       5 | Sport
      47 | Monitored non-volatile initiative                  |       3 |       5 | Sport
      48 | Pre-emptive next generation infrastructure         |       3 |       5 | Sport
      49 | Switchable 5th generation parallelism              |       4 |       5 | Sport
      50 | Adaptive modular approach                          |       2 |       5 | Sport
      51 | Synergistic zero defect info-mediaries             |       2 |       5 | Sport
      52 | Persevering empowering customer loyalty            |       3 |       5 | Sport
      53 | Robust foreground leverage                         |       1 |       5 | Sport
      54 | Customizable cohesive capacity                     |       6 |       5 | Sport
      55 | Progressive modular archive                        |       3 |       5 | Sport
      56 | Reduced fresh-thinking process improvement         |       2 |       5 | Sport
      57 | Seamless optimal leverage                          |       6 |       5 | Sport
      58 | Universal exuding protocol                         |       5 |       5 | Sport
      59 | Realigned client-driven database                   |       6 |       5 | Sport
      60 | Balanced hybrid portal                             |       5 |       5 | Sport
      61 | Customizable well-modulated encryption             |       5 |       5 | Sport
      62 | Cross-group reciprocal firmware                    |       3 |       5 | Sport
      63 | 4th generation Graphical User Interface            |       4 |       5 | Sport
      64 | Business-focused solution-oriented moratorium      |       5 |       5 | Sport
      65 | Synergistic scalable capability                    |       5 |       5 | Sport
      66 | Distributed uniform Graphic Interface              |       5 |       5 | Sport
      67 | Stand-alone composite Graphical User Interface     |       2 |       5 | Sport
      68 | Future-proofed leading edge customer loyalty       |       4 |       5 | Sport
      69 | Profound human-resource forecast                   |       6 |       5 | Sport
      70 | Advanced neutral portal                            |       3 |       5 | Sport
      71 | Customer-focused needs-based protocol              |       3 |       5 | Sport
      72 | User-friendly encompassing array                   |       6 |       5 | Sport
      73 | Decentralized human-resource infrastructure        |       2 |       5 | Sport
      74 | Balanced modular website                           |       2 |       5 | Sport
      75 | Horizontal explicit benchmark                      |       2 |       5 | Sport
      76 | Re-engineered 24/7 knowledge base                  |       1 |       5 | Sport
      77 | Innovative web-enabled extranet                    |       2 |       5 | Sport
      78 | Exclusive analyzing open architecture              |       2 |       5 | Sport
      79 | Fundamental global archive                         |       3 |       5 | Sport
      80 | Profound value-added intranet                      |       5 |       5 | Sport
      81 | Networked global open system                       |       6 |       5 | Sport
      82 | Persistent demand-driven complexity                |       5 |       5 | Sport
      83 | Focused secondary initiative                       |       5 |       5 | Sport
      84 | Digitized tertiary groupware                       |       3 |       5 | Sport
      85 | Enhanced homogeneous paradigm                      |       4 |       5 | Sport
      86 | Inverse high-level attitude                        |       4 |       5 | Sport
      87 | Quality-focused upward-trending throughput         |       4 |       5 | Sport
```

```
      88 | Team-oriented stable project                       |      6 |      5 | Sport
      89 | Total intangible artificial intelligence          |      3 |      5 | Sport
      90 | Streamlined asynchronous functionalities          |      5 |      5 | Sport
      91 | Vision-oriented attitude-oriented core            |      5 |      5 | Sport
      92 | Integrated 24/7 interface                         |      2 |      5 | Sport
      93 | Advanced didactic Graphic Interface               |      1 |      5 | Sport
      94 | Exclusive multimedia middleware                   |      6 |      5 | Sport
      95 | Ameliorated next generation orchestration         |      6 |      5 | Sport
      96 | Front-line demand-driven utilisation              |      5 |      5 | Sport
      97 | Organic clear-thinking system engine              |      3 |      5 | Sport
      98 | Persistent incremental model                      |      3 |      5 | Sport
      99 | Ergonomic solution-oriented local area network    |      2 |      5 | Sport
     100 | Robust mission-critical complexity                |      5 |      5 | Sport
       1 | Multi-layered multi-tasking initiative            |      2 |      6 | Health
       2 | Operative analyzing task-force                    |      1 |      6 | Health
       3 | Exclusive client-server array                     |      5 |      6 | Health
       4 | Balanced client-server product                    |      6 |      6 | Health
       5 | Exclusive background website                      |      5 |      6 | Health
       6 | Pre-emptive holistic intranet                     |      6 |      6 | Health
       7 | Re-engineered cohesive methodology                |      1 |      6 | Health
       8 | Robust directional projection                     |      2 |      6 | Health
       9 | Inverse transitional infrastructure               |      4 |      6 | Health
      10 | Multi-tiered explicit paradigm                    |      6 |      6 | Health
      11 | Re-engineered explicit software                   |      2 |      6 | Health
      12 | Cross-platform fresh-thinking core                |      3 |      6 | Health
      13 | Diverse neutral emulation                         |      4 |      6 | Health
      14 | Up-sized composite challenge                      |      4 |      6 | Health
      15 | Intuitive directional complexity                  |      4 |      6 | Health
      16 | Universal encompassing conglomeration             |      5 |      6 | Health
      17 | Multi-channelled well-modulated analyzer          |      2 |      6 | Health
      18 | Re-engineered actuating capability                |      4 |      6 | Health
      19 | Public-key interactive encoding                   |      2 |      6 | Health
      20 | Monitored asynchronous function                   |      6 |      6 | Health
      21 | Proactive methodical data-warehouse               |      4 |      6 | Health
      22 | Balanced real-time info-mediaries                 |      1 |      6 | Health
      23 | Right-sized mission-critical pricing structure    |      6 |      6 | Health
      24 | Synergistic homogeneous ability                   |      5 |      6 | Health
      25 | Open-source impactful archive                     |      5 |      6 | Health
      26 | Realigned 5th generation artificial intelligence  |      2 |      6 | Health
      27 | Configurable methodical firmware                  |      5 |      6 | Health
      28 | Profound optimal encryption                       |      3 |      6 | Health
      29 | Vision-oriented user-facing framework             |      2 |      6 | Health
      30 | Secured holistic hierarchy                        |      2 |      6 | Health
      31 | Assimilated regional instruction set              |      2 |      6 | Health
      32 | Business-focused holistic help-desk               |      3 |      6 | Health
      33 | Virtual stable Graphic Interface                  |      5 |      6 | Health
      34 | Implemented optimizing benchmark                  |      1 |      6 | Health
      35 | Adaptive static website                           |      1 |      6 | Health
      36 | Virtual impactful success                         |      2 |      6 | Health
      37 | Open-architected homogeneous concept              |      6 |      6 | Health
      38 | Diverse reciprocal knowledge base                 |      1 |      6 | Health
      39 | Realigned homogeneous hub                         |      5 |      6 | Health
      40 | Switchable tangible product                       |      4 |      6 | Health
      41 | Universal global hub                              |      2 |      6 | Health
      42 | Enhanced discrete function                        |      4 |      6 | Health
      43 | Horizontal asynchronous intranet                  |      4 |      6 | Health
      44 | Quality-focused foreground analyzer               |      5 |      6 | Health
      45 | Configurable analyzing solution                   |      3 |      6 | Health
      46 | Fully-configurable full-range interface           |      6 |      6 | Health
      47 | Monitored non-volatile initiative                 |      3 |      6 | Health
      48 | Pre-emptive next generation infrastructure        |      3 |      6 | Health
      49 | Switchable 5th generation parallelism             |      4 |      6 | Health
      50 | Adaptive modular approach                         |      2 |      6 | Health
      51 | Synergistic zero defect info-mediaries            |      2 |      6 | Health
      52 | Persevering empowering customer loyalty           |      3 |      6 | Health
      53 | Robust foreground leverage                        |      1 |      6 | Health
      54 | Customizable cohesive capacity                    |      6 |      6 | Health
      55 | Progressive modular archive                       |      3 |      6 | Health
      56 | Reduced fresh-thinking process improvement        |      2 |      6 | Health
      57 | Seamless optimal leverage                         |      6 |      6 | Health
      58 | Universal exuding protocol                        |      5 |      6 | Health
      59 | Realigned client-driven database                  |      6 |      6 | Health
      60 | Balanced hybrid portal                            |      5 |      6 | Health
      61 | Customizable well-modulated encryption            |      5 |      6 | Health
      62 | Cross-group reciprocal firmware                   |      3 |      6 | Health
```

```
      63 | 4th generation Graphical User Interface          |        4 |       6 | Health
      64 | Business-focused solution-oriented moratorium    |        5 |       6 | Health
      65 | Synergistic scalable capability                  |        5 |       6 | Health
      66 | Distributed uniform Graphic Interface            |        5 |       6 | Health
      67 | Stand-alone composite Graphical User Interface   |        2 |       6 | Health
      68 | Future-proofed leading edge customer loyalty     |        4 |       6 | Health
      69 | Profound human-resource forecast                 |        6 |       6 | Health
      70 | Advanced neutral portal                          |        3 |       6 | Health
      71 | Customer-focused needs-based protocol            |        3 |       6 | Health
      72 | User-friendly encompassing array                 |        6 |       6 | Health
      73 | Decentralized human-resource infrastructure      |        2 |       6 | Health
      74 | Balanced modular website                         |        2 |       6 | Health
      75 | Horizontal explicit benchmark                    |        2 |       6 | Health
      76 | Re-engineered 24/7 knowledge base                |        1 |       6 | Health
      77 | Innovative web-enabled extranet                  |        2 |       6 | Health
      78 | Exclusive analyzing open architecture            |        2 |       6 | Health
      79 | Fundamental global archive                       |        3 |       6 | Health
      80 | Profound value-added intranet                    |        5 |       6 | Health
      81 | Networked global open system                     |        6 |       6 | Health
      82 | Persistent demand-driven complexity              |        5 |       6 | Health
      83 | Focused secondary initiative                     |        5 |       6 | Health
      84 | Digitized tertiary groupware                     |        3 |       6 | Health
      85 | Enhanced homogeneous paradigm                    |        4 |       6 | Health
      86 | Inverse high-level attitude                      |        4 |       6 | Health
      87 | Quality-focused upward-trending throughput       |        4 |       6 | Health
      88 | Team-oriented stable project                     |        6 |       6 | Health
      89 | Total intangible artificial intelligence         |        3 |       6 | Health
      90 | Streamlined asynchronous functionalities         |        5 |       6 | Health
      91 | Vision-oriented attitude-oriented core           |        5 |       6 | Health
      92 | Integrated 24/7 interface                        |        2 |       6 | Health
      93 | Advanced didactic Graphic Interface              |        1 |       6 | Health
      94 | Exclusive multimedia middleware                  |        6 |       6 | Health
      95 | Ameliorated next generation orchestration        |        6 |       6 | Health
      96 | Front-line demand-driven utilisation             |        5 |       6 | Health
      97 | Organic clear-thinking system engine             |        3 |       6 | Health
      98 | Persistent incremental model                     |        3 |       6 | Health
      99 | Ergonomic solution-oriented local area network   |        2 |       6 | Health
     100 | Robust mission-critical complexity               |        5 |       6 | Health
(600 rows)
```

15. Look at the printout for the question above and find the category of the product "Multi-layered multi-tasking initiative"

16. Use the following command to narrow down the search

LANGUAGE: SQL

```sql
select * from category, product where prod_name = 'Multi-layered multi-tasking initiative'
    ↪ ;
```

When we don't join tables properly, the output we are given is called a 'Cartesian Product'. This is bad.

17. Run the following code

LANGUAGE: SQL

```sql
select * from category
join product on category.cat_id = product.prod_cat;
```

LANGUAGE: Unknown

```
 cat_id |  cat_name   | prod_id |                    prod_name                     |
    ↪ prod_cat
--------+-------------+---------+--------------------------------------------------+----------
    ↪
      2 | Ladies Wear |       1 | Multi-layered multi-tasking initiative           |
```

```
    ↪   2
  1 | Men's Wear  |       2 | Operative analyzing task-force                    |
    ↪   1
  5 | Sport       |       3 | Exclusive client-server array                     |
    ↪   5
  6 | Health      |       4 | Balanced client-server product                    |
    ↪   6
  5 | Sport       |       5 | Exclusive background website                      |
    ↪   5
  6 | Health      |       6 | Pre-emptive holistic intranet                     |
    ↪   6
  1 | Men's Wear  |       7 | Re-engineered cohesive methodology                |
    ↪   1
  2 | Ladies Wear |       8 | Robust directional projection                     |
    ↪   2
  4 | Outdoor     |       9 | Inverse transitional infrastructure               |
    ↪   4
  6 | Health      |      10 | Multi-tiered explicit paradigm                    |
    ↪   6
  2 | Ladies Wear |      11 | Re-engineered explicit software                   |
    ↪   2
  3 | Kid's Wear  |      12 | Cross-platform fresh-thinking core                |
    ↪   3
  4 | Outdoor     |      13 | Diverse neutral emulation                         |
    ↪   4
  4 | Outdoor     |      14 | Up-sized composite challenge                      |
    ↪   4
  4 | Outdoor     |      15 | Intuitive directional complexity                  |
    ↪   4
  5 | Sport       |      16 | Universal encompassing conglomeration             |
    ↪   5
  2 | Ladies Wear |      17 | Multi-channelled well-modulated analyzer          |
    ↪   2
  4 | Outdoor     |      18 | Re-engineered actuating capability                |
    ↪   4
  2 | Ladies Wear |      19 | Public-key interactive encoding                   |
    ↪   2
  6 | Health      |      20 | Monitored asynchronous function                   |
    ↪   6
  4 | Outdoor     |      21 | Proactive methodical data-warehouse               |
    ↪   4
  1 | Men's Wear  |      22 | Balanced real-time info-mediaries                 |
    ↪   1
  6 | Health      |      23 | Right-sized mission-critical pricing structure    |
    ↪   6
  5 | Sport       |      24 | Synergistic homogeneous ability                   |
    ↪   5
  5 | Sport       |      25 | Open-source impactful archive                     |
    ↪   5
  2 | Ladies Wear |      26 | Realigned 5th generation artificial intelligence  |
    ↪   2
  5 | Sport       |      27 | Configurable methodical firmware                  |
    ↪   5
  3 | Kid's Wear  |      28 | Profound optimal encryption                       |
    ↪   3
  2 | Ladies Wear |      29 | Vision-oriented user-facing framework             |
    ↪   2
  2 | Ladies Wear |      30 | Secured holistic hierarchy                        |
    ↪   2
  2 | Ladies Wear |      31 | Assimilated regional instruction set              |
    ↪   2
  3 | Kid's Wear  |      32 | Business-focused holistic help-desk               |
    ↪   3
  5 | Sport       |      33 | Virtual stable Graphic Interface                  |
    ↪   5
  1 | Men's Wear  |      34 | Implemented optimizing benchmark                  |
    ↪   1
  1 | Men's Wear  |      35 | Adaptive static website                           |
    ↪   1
  2 | Ladies Wear |      36 | Virtual impactful success                         |
    ↪   2
  6 | Health      |      37 | Open-architected homogeneous concept              |
    ↪   6
  1 | Men's Wear  |      38 | Diverse reciprocal knowledge base                 |
    ↪   1
```

```
    5 | Sport       |      39 | Realigned homogeneous hub                          |
    ↪   5
    4 | Outdoor     |      40 | Switchable tangible product                        |
    ↪   4
    2 | Ladies Wear |      41 | Universal global hub                               |
    ↪   2
    4 | Outdoor     |      42 | Enhanced discrete function                         |
    ↪   4
    4 | Outdoor     |      43 | Horizontal asynchronous intranet                   |
    ↪   4
    5 | Sport       |      44 | Quality-focused foreground analyzer                |
    ↪   5
    3 | Kid's Wear  |      45 | Configurable analyzing solution                    |
    ↪   3
    6 | Health      |      46 | Fully-configurable full-range interface            |
    ↪   6
    3 | Kid's Wear  |      47 | Monitored non-volatile initiative                  |
    ↪   3
    3 | Kid's Wear  |      48 | Pre-emptive next generation infrastructure         |
    ↪   3
    4 | Outdoor     |      49 | Switchable 5th generation parallelism              |
    ↪   4
    2 | Ladies Wear |      50 | Adaptive modular approach                          |
    ↪   2
    2 | Ladies Wear |      51 | Synergistic zero defect info-mediaries             |
    ↪   2
    3 | Kid's Wear  |      52 | Persevering empowering customer loyalty            |
    ↪   3
    1 | Men's Wear  |      53 | Robust foreground leverage                         |
    ↪   1
    6 | Health      |      54 | Customizable cohesive capacity                     |
    ↪   6
    3 | Kid's Wear  |      55 | Progressive modular archive                        |
    ↪   3
    2 | Ladies Wear |      56 | Reduced fresh-thinking process improvement         |
    ↪   2
    6 | Health      |      57 | Seamless optimal leverage                          |
    ↪   6
    5 | Sport       |      58 | Universal exuding protocol                         |
    ↪   5
    6 | Health      |      59 | Realigned client-driven database                   |
    ↪   6
    5 | Sport       |      60 | Balanced hybrid portal                             |
    ↪   5
    5 | Sport       |      61 | Customizable well-modulated encryption             |
    ↪   5
    3 | Kid's Wear  |      62 | Cross-group reciprocal firmware                    |
    ↪   3
    4 | Outdoor     |      63 | 4th generation Graphical User Interface            |
    ↪   4
    5 | Sport       |      64 | Business-focused solution-oriented moratorium      |
    ↪   5
    5 | Sport       |      65 | Synergistic scalable capability                    |
    ↪   5
    5 | Sport       |      66 | Distributed uniform Graphic Interface              |
    ↪   5
    2 | Ladies Wear |      67 | Stand-alone composite Graphical User Interface     |
    ↪   2
    4 | Outdoor     |      68 | Future-proofed leading edge customer loyalty       |
    ↪   4
    6 | Health      |      69 | Profound human-resource forecast                   |
    ↪   6
    3 | Kid's Wear  |      70 | Advanced neutral portal                            |
    ↪   3
    3 | Kid's Wear  |      71 | Customer-focused needs-based protocol              |
    ↪   3
    6 | Health      |      72 | User-friendly encompassing array                   |
    ↪   6
    2 | Ladies Wear |      73 | Decentralized human-resource infrastructure        |
    ↪   2
    2 | Ladies Wear |      74 | Balanced modular website                           |
    ↪   2
    2 | Ladies Wear |      75 | Horizontal explicit benchmark                      |
    ↪   2
    1 | Men's Wear  |      76 | Re-engineered 24/7 knowledge base                  |
```

```
    ↪   1
  2 | Ladies Wear |      77 | Innovative web-enabled extranet                    |
    ↪   2
  2 | Ladies Wear |      78 | Exclusive analyzing open architecture              |
    ↪   2
  3 | Kid's Wear  |      79 | Fundamental global archive                         |
    ↪   3
  5 | Sport       |      80 | Profound value-added intranet                      |
    ↪   5
  6 | Health      |      81 | Networked global open system                       |
    ↪   6
  5 | Sport       |      82 | Persistent demand-driven complexity                |
    ↪   5
  5 | Sport       |      83 | Focused secondary initiative                       |
    ↪   5
  3 | Kid's Wear  |      84 | Digitized tertiary groupware                       |
    ↪   3
  4 | Outdoor     |      85 | Enhanced homogeneous paradigm                      |
    ↪   4
  4 | Outdoor     |      86 | Inverse high-level attitude                        |
    ↪   4
  4 | Outdoor     |      87 | Quality-focused upward-trending throughput         |
    ↪   4
  6 | Health      |      88 | Team-oriented stable project                       |
    ↪   6
  3 | Kid's Wear  |      89 | Total intangible artificial intelligence           |
    ↪   3
  5 | Sport       |      90 | Streamlined asynchronous functionalities           |
    ↪   5
  5 | Sport       |      91 | Vision-oriented attitude-oriented core             |
    ↪   5
  2 | Ladies Wear |      92 | Integrated 24/7 interface                          |
    ↪   2
  1 | Men's Wear  |      93 | Advanced didactic Graphic Interface                |
    ↪   1
  6 | Health      |      94 | Exclusive multimedia middleware                    |
    ↪   6
  6 | Health      |      95 | Ameliorated next generation orchestration          |
    ↪   6
  5 | Sport       |      96 | Front-line demand-driven utilisation               |
    ↪   5
  3 | Kid's Wear  |      97 | Organic clear-thinking system engine               |
    ↪   3
  3 | Kid's Wear  |      98 | Persistent incremental model                       |
    ↪   3
  2 | Ladies Wear |      99 | Ergonomic solution-oriented local area network     |
    ↪   2
  5 | Sport       |     100 | Robust mission-critical complexity                 |
    ↪   5
(100 rows)
```

18. How many rows are returned now.
    100

19. Write the code to find the category information for the product ”Multi-layered multi-tasking initiative”

LANGUAGE: SQL

```sql
select * from category
join product on category.cat_id = product.prod_cat
where prod_name = 'Multi-layered multi-tasking initiative';
```

LANGUAGE: Unknown

```
 cat_id |  cat_name   | prod_id |               prod_name                | prod_cat
--------+-------------+---------+----------------------------------------+----------
      2 | Ladies Wear |       1 | Multi-layered multi-tasking initiative |        2
(1 row)
```

20. Run the following code

```
LANGUAGE: SQL
select count(*) from customer, cust_order;
```

This will connect every customer to every order stored in the `cust_order` table.

```
LANGUAGE: Unknown
 count
-------
  1650
(1 row)
```

21. Write a query that will display the customer's first name, their last name and the order numbers, stored in the `cust_order` table as the `cust_ord_id`, but only for the customer with the `cust_id` of 1. Copy the code and the printout below.

```
LANGUAGE: SQL
select customer.cust_fname, customer.cust_lname, cust_order.cust_ord_id from customer
join cust_order on customer.cust_id = cust_order.cust_id
where cust_order.cust_id = 1;
```

```
LANGUAGE: Unknown
 cust_fname | cust_lname | cust_ord_id
------------+------------+-------------
 Jobey      | Boeter     |          26
 Jobey      | Boeter     |          34
 Jobey      | Boeter     |          39
 Jobey      | Boeter     |          57
 Jobey      | Boeter     |          68
 Jobey      | Boeter     |          71
 Jobey      | Boeter     |          77
 Jobey      | Boeter     |          91
 Jobey      | Boeter     |          98
 Jobey      | Boeter     |          99
 Jobey      | Boeter     |         131
 Jobey      | Boeter     |         143
 Jobey      | Boeter     |         146
(13 rows)
```

22. Now try to see if you can add the `staff_fname`, the `staff_lname` to the above printout. You will need to join the `staff` table. Look at the ERD and the printout from to find the matching primary key and foreign key

```
LANGUAGE: SQL
select customer.cust_fname, customer.cust_lname, cust_order.cust_ord_id, staff.staff_fname
    ↪ , staff.staff_lname from customer
join cust_order on customer.cust_id = cust_order.cust_id
join staff on cust_order.staff_id = staff.staff_id
where cust_order.cust_id = 1;
```

```
LANGUAGE: Unknown
 cust_fname | cust_lname | cust_ord_id | staff_fname | staff_lname
------------+------------+-------------+-------------+-------------
 Jobey      | Boeter     |          26 | Montgomery  | Housegoe
 Jobey      | Boeter     |          34 | Hanan       | Gloster
```

```
 Jobey      | Boeter     |          39 | Hanan       | Gloster
 Jobey      | Boeter     |          57 | Nikoletta   | Shrimpton
 Jobey      | Boeter     |          68 | Montgomery  | Housegoe
 Jobey      | Boeter     |          71 | Nikoletta   | Shrimpton
 Jobey      | Boeter     |          77 | Hanan       | Gloster
 Jobey      | Boeter     |          91 | Niel        | Welsby
 Jobey      | Boeter     |          98 | Montgomery  | Housegoe
 Jobey      | Boeter     |          99 | Janeva      | Gillicuddy
 Jobey      | Boeter     |         131 | Aura        | Clewlowe
 Jobey      | Boeter     |         143 | Janeva      | Gillicuddy
 Jobey      | Boeter     |         146 | Montgomery  | Housegoe
(13 rows)
```

23. If you have got this far, try to get a printout that joins the `role` table, the `staff` table, the `cust_order` table and the `customer` table. Retrieve the roles of anyone who has worked on an order for `cust_id` of 4.

LANGUAGE: SQL

```sql
select customer.cust_fname, customer.cust_lname, cust_order.cust_ord_id, staff.staff_fname
    ↪ , staff.staff_lname, role.role_id, role.role_name from customer
join cust_order on customer.cust_id = cust_order.cust_id
join staff on cust_order.staff_id = staff.staff_id
join role on staff.role = role.role_id
where customer.cust_id = 4;
```

LANGUAGE: Unknown

```
 cust_fname |    cust_lname    | cust_ord_id | staff_fname | staff_lname | role_id |
    ↪ role_name
------------+------------------+-------------+-------------+-------------+---------+-----------------
    ↪
 Chadd      | Franz-Schoninger |           1 | Aura        | Clewlowe    |       3 | Post
    ↪ Sales
 Chadd      | Franz-Schoninger |           7 | Aura        | Clewlowe    |       3 | Post
    ↪ Sales
 Chadd      | Franz-Schoninger |          66 | Montgomery  | Housegoe    |       1 | Order
    ↪  Picker
 Chadd      | Franz-Schoninger |          81 | Janeva      | Gillicuddy  |       5 | Misc
 Chadd      | Franz-Schoninger |          93 | Niel        | Welsby      |       2 | Final
    ↪  Packer
 Chadd      | Franz-Schoninger |          97 | Aura        | Clewlowe    |       3 | Post
    ↪ Sales
 Chadd      | Franz-Schoninger |         107 | Hanan       | Gloster     |       4 |
    ↪ Customer Retain
 Chadd      | Franz-Schoninger |         109 | Nikoletta   | Shrimpton   |       4 |
    ↪ Customer Retain
 Chadd      | Franz-Schoninger |         124 | Aura        | Clewlowe    |       3 | Post
    ↪ Sales
 Chadd      | Franz-Schoninger |         129 | Nikoletta   | Shrimpton   |       4 |
    ↪ Customer Retain
(10 rows)
```

S.12. Joins and Narrowing Focus

17-11-22 13:00 Mark RB LT1

Introduction to Joins

Joins are key to understanding how to get useful information out of a database. Data in an individual table is of limited use, to get good data, we need to join multiple tables together. We might only want some information.
To get these individual items from one table, we can do this with

```sql
SELECT firstName, lastName, emailAddress from TABLE;
```

However, this will still return every record.
We can narrow this down, using the `WHERE=condition` clause. For example,

```sql
SELECT firstName, lastName, emailAddress WHERE town = 'Portsmouth';
```

This will give us all the records where the town attribute is equal to Portsmouth
What if we want to get data from multiple tables? Here we have to use Joins.

Joins

To create a join between two tables, one table needs to have a foreign key where that is the primary key in the other table you wish to join.
When creating joins between tables, it's important to ensure that the correct attributes in each tables are joined. Just because an result is produced form the query, it doesn't necessarily mean its the right one.
The data types between the two attributes which are being joined have to match whilst the names used in each table do not.

Cartesian Product

This is the result of a wrong join.
It is where every single record in one table is joined to every single table in another table. For example, two tables: customer and order. Customer has 11 records and order has 150. 150×11 gives 1650 rows as output. This provides a big problem when attempting to join two big tables together.

The Correct Way

When joining two tables correctly, we have to tell the DMBS what values match.

```sql
SELECT CUSTOMER.CUST_ID, CUST_ORD_ID FROM CUSTOMER JOIN cust_order ON CUSTOMER.CUST_ID =
    ↪ CUST_ORDER.CUST_ID;
```

Query above returns 150 rows of data. We know this is correct as it is the same as the number of rows in orders table.

Another Correct Way

We do not have to use the `join` keyword, instead we can use the `WHERE` condition.

```
LANGUAGE: SQL
1 SELECT CUST_LNAME, CUST_ORD_ID FROM CUSTOMER, CUST_ORDER WHERE CUSTOMER.CUST_ID = CUST_ORDER.
    ↪ CUST_ID;
```

This will happily produce 150 rows.
To join more than two tables, we have to use an `AND` statement in the `WHERE` condition.