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. More Joins

01-12-22 13:00 Mark RB LT1

The joins we have looked at so far are `inner` joins. This displays the data where the tables overlap. For example

LANGUAGE: SQL

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

Will probably use this the most.

Left Join

This will produce everything form the left table (`customer`) and the overlapping data from the right hand table (`cust_order`) where there is a match on the common attribute to both (`cust_id`)

LANGUAGE: SQL

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

Right Join

This will return everything from the right table (`cust_order`) and common data where it is there.

LANGUAGE: SQL

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

It is important to use the correct join for the situation as when used incorrectly as you won't get the data returned which you are expecting.

Outer Joins

This gives everything from all the tables mentioned in the query.

LANGUAGE: SQL

```sql
SELECT role_name, staff_lname, staff_fname FROM staff FULL OUTER JOIN
ROLE ON ROLE=role_id;
```

Will probably use this the least.

Things To Remember

- Use the correct type of join for the job
- Match like for like

S.2. PRACTICAL: FURTHER JOINS

01-12-22 14:00 Mark etc FTC 3

Tutor Led

We need to insert two more roles into the `Role` table.

LANGUAGE: SQL

```sql
INSERT INTO ROLE (role_name)
VALUES ('Cleaner');

INSERT INTO ROLE (role_name)
VALUES ('Pre Sales');
```

Then run the following.

LANGUAGE: SQL

```sql
SELECT count(*)
FROM ROLE;
```

This generates the following output

LANGUAGE: Unknown

```
 count
-------
     7
(1 row)
```

Student Tasks

1. Write a query that correctly displays the staff members first and last names, their email addresses and their roles. Use the method that uses the `JOIN` keyword. Copy the code and answer below.

LANGUAGE: SQL

```sql
SELECT staff.staff_fname, staff.staff_lname, staff.home_email, role.role_name FROM staff
JOIN role on staff.role = role.role_id;
```

LANGUAGE: Unknown

```
 staff_fname | staff_lname |         home_email          |    role_name
-------------+-------------+-----------------------------+-----------------
 Montgomery  | Housegoe    | mhousegoe2@ucoz.ru          | Order Picker
 Niel        | Welsby      | nwelsby0@rambler.ru         | Final Packer
 Jillene     | Revitt      | jrevitt8@cornell.edu        | Post Sales
 Harriette   | Fewster     | hfewster7@independent.co.uk | Post Sales
 Aura        | Clewlowe    | aclewlowe5@google.com.au    | Post Sales
 Hanan       | Gloster     | hgloster3@blogger.com       | Customer Retain
 Nikoletta   | Shrimpton   | nshrimpton1@unblog.fr       | Customer Retain
 Tim         | Illem       | tillem9@dedecms.com         | Misc
 Nell        | Olsson      | nolsson6@jiathis.com        | Misc
 Janeva      | Gillicuddy  | jgillicuddy4@altervista.org | Misc
(10 rows)
```

2. Rewrite the query created in 1 but this time use the `WHERE` keyword. Copy the code and answer below.

LANGUAGE: SQL

```sql
SELECT staff.staff_fname, staff.staff_lname, staff.home_email, role.role_name FROM staff, role
WHERE staff.role = role.role_id;
```

LANGUAGE: Unknown

```
 staff_fname | staff_lname |         home_email          |    role_name
-------------+-------------+-----------------------------+-----------------
 Montgomery  | Housegoe    | mhousegoe2@ucoz.ru          | Order Picker
 Niel        | Welsby      | nwelsby0@rambler.ru         | Final Packer
 Jillene     | Revitt      | jrevitt8@cornell.edu        | Post Sales
 Harriette   | Fewster     | hfewster7@independent.co.uk | Post Sales
 Aura        | Clewlowe    | aclewlowe5@google.com.au    | Post Sales
 Hanan       | Gloster     | hgloster3@blogger.com       | Customer Retain
 Nikoletta   | Shrimpton   | nshrimpton1@unblog.fr       | Customer Retain
 Tim         | Illem       | tillem9@dedecms.com         | Misc
 Nell        | Olsson      | nolsson6@jiathis.com        | Misc
 Janeva      | Gillicuddy  | jgillicuddy4@altervista.org | Misc
(10 rows)
```

3. List the customer first and last names with their email addresses and the product names of the products they have ordered. But only for the customers who live in Waekolong. Copy the code and the answer below.

LANGUAGE: SQL

```sql
SELECT customer.cust_fname, customer.cust_lname, customer.email, product.prod_name FROM customer
JOIN cust_order ON customer.cust_id=cust_order.cust_id
JOIN manifest ON cust_order.cust_ord_id=manifest.cust_ord_id
JOIN product on manifest.prod_id=product.prod_id
WHERE customer.town='Waekolong';
```

LANGUAGE: Unknown

```
   cust_fname     | cust_lname |          email           |                    prod_name
------------------+------------+--------------------------+-------------------------------------------------
 Marie-françoise | Currier    | acurrier0@economist.com | Vision-oriented attitude-oriented core
 Marie-françoise | Currier    | acurrier0@economist.com | Balanced client-server product
 Marie-françoise | Currier    | acurrier0@economist.com | Exclusive client-server array
 Marie-françoise | Currier    | acurrier0@economist.com | Universal encompassing conglomeration
 Marie-françoise | Currier    | acurrier0@economist.com | Synergistic homogeneous ability
 Marie-françoise | Currier    | acurrier0@economist.com | Universal exuding protocol
 Marie-françoise | Currier    | acurrier0@economist.com | Universal global hub
 Marie-françoise | Currier    | acurrier0@economist.com | Balanced real-time info-mediaries
 Marie-françoise | Currier    | acurrier0@economist.com | Integrated 24/7 interface
 Marie-françoise | Currier    | acurrier0@economist.com | Re-engineered explicit software
 Marie-françoise | Currier    | acurrier0@economist.com | Customizable cohesive capacity
 Marie-françoise | Currier    | acurrier0@economist.com | Robust mission-critical complexity
 Marie-françoise | Currier    | acurrier0@economist.com | Organic clear-thinking system engine
 Marie-françoise | Currier    | acurrier0@economist.com | Stand-alone composite Graphical User Interface
(14 rows)
```

4. Write a query that returns all categories and the product names and order the output into category order. Copy the code and the answer below.

LANGUAGE: SQL

```sql
SELECT category.cat_name, product.prod_name FROM category
JOIN product ON product.prod_cat = category.cat_id
ORDER BY category.cat_name;
```

LANGUAGE: Unknown

```
  cat_name   |                    prod_name
-------------+---------------------------------------------------
 Health      | Exclusive multimedia middleware
 Health      | Pre-emptive holistic intranet
 Health      | Ameliorated next generation orchestration
 Health      | Monitored asynchronous function
 Health      | Right-sized mission-critical pricing structure
 Health      | Profound human-resource forecast
 Health      | Realigned client-driven database
 Health      | Seamless optimal leverage
 Health      | User-friendly encompassing array
 Health      | Customizable cohesive capacity
 Health      | Fully-configurable full-range interface
 Health      | Team-oriented stable project
 Health      | Multi-tiered explicit paradigm
 Health      | Balanced client-server product
 Health      | Open-architected homogeneous concept
 Health      | Networked global open system
 Kid's Wear  | Persistent incremental model
 Kid's Wear  | Cross-platform fresh-thinking core
 Kid's Wear  | Advanced neutral portal
 Kid's Wear  | Customer-focused needs-based protocol
 Kid's Wear  | Organic clear-thinking system engine
 Kid's Wear  | Profound optimal encryption
 Kid's Wear  | Business-focused holistic help-desk
 Kid's Wear  | Total intangible artificial intelligence
 Kid's Wear  | Configurable analyzing solution
 Kid's Wear  | Monitored non-volatile initiative
 Kid's Wear  | Pre-emptive next generation infrastructure
 Kid's Wear  | Persevering empowering customer loyalty
 Kid's Wear  | Progressive modular archive
 Kid's Wear  | Digitized tertiary groupware
 Kid's Wear  | Fundamental global archive
 Kid's Wear  | Cross-group reciprocal firmware
 Ladies Wear | Decentralized human-resource infrastructure
 Ladies Wear | Adaptive modular approach
 Ladies Wear | Synergistic zero defect info-mediaries
 Ladies Wear | Public-key interactive encoding
 Ladies Wear | Multi-channelled well-modulated analyzer
 Ladies Wear | Realigned 5th generation artificial intelligence
 Ladies Wear | Vision-oriented user-facing framework
 Ladies Wear | Secured holistic hierarchy
 Ladies Wear | Assimilated regional instruction set
 Ladies Wear | Integrated 24/7 interface
 Ladies Wear | Virtual impactful success
 Ladies Wear | Exclusive analyzing open architecture
 Ladies Wear | Innovative web-enabled extranet
 Ladies Wear | Robust directional projection
 Ladies Wear | Universal global hub
 Ladies Wear | Ergonomic solution-oriented local area network
 Ladies Wear | Horizontal explicit benchmark
 Ladies Wear | Reduced fresh-thinking process improvement
 Ladies Wear | Balanced modular website
 Ladies Wear | Stand-alone composite Graphical User Interface
 Ladies Wear | Multi-layered multi-tasking initiative
 Ladies Wear | Re-engineered explicit software
 Men's Wear  | Implemented optimizing benchmark
 Men's Wear  | Adaptive static website
 Men's Wear  | Balanced real-time info-mediaries
 Men's Wear  | Re-engineered cohesive methodology
 Men's Wear  | Diverse reciprocal knowledge base
 Men's Wear  | Robust foreground leverage
 Men's Wear  | Advanced didactic Graphic Interface
 Men's Wear  | Re-engineered 24/7 knowledge base
 Men's Wear  | Operative analyzing task-force
 Outdoor      | 4th generation Graphical User Interface
 Outdoor      | Inverse transitional infrastructure
 Outdoor      | Diverse neutral emulation
 Outdoor      | Up-sized composite challenge
 Outdoor      | Intuitive directional complexity
 Outdoor      | Re-engineered actuating capability
 Outdoor      | Proactive methodical data-warehouse
 Outdoor      | Switchable tangible product
 Outdoor      | Enhanced discrete function
 Outdoor      | Horizontal asynchronous intranet
 Outdoor      | Switchable 5th generation parallelism
 Outdoor      | Future-proofed leading edge customer loyalty
 Outdoor      | Enhanced homogeneous paradigm
 Outdoor      | Inverse high-level attitude
 Outdoor      | Quality-focused upward-trending throughput
 Sport        | Customizable well-modulated encryption
 Sport        | Profound value-added intranet
 Sport        | Balanced hybrid portal
 Sport        | Persistent demand-driven complexity
 Sport        | Focused secondary initiative
 Sport        | Universal exuding protocol
 Sport        | Exclusive background website
 Sport        | Exclusive client-server array
 Sport        | Robust mission-critical complexity
 Sport        | Quality-focused foreground analyzer
 Sport        | Realigned homogeneous hub
 Sport        | Streamlined asynchronous functionalities
 Sport        | Vision-oriented attitude-oriented core
 Sport        | Virtual stable Graphic Interface
 Sport        | Configurable methodical firmware
 Sport        | Open-source impactful archive
 Sport        | Synergistic homogeneous ability
 Sport        | Front-line demand-driven utilisation
 Sport        | Universal encompassing conglomeration
 Sport        | Distributed uniform Graphic Interface
 Sport        | Synergistic scalable capability
 Sport        | Business-focused solution-oriented moratorium
(100 rows)
```

5. Rewrite the query for Q4 so that the output is ordered by category, then the product id. Copy the code and the answer below.

LANGUAGE: SQL

```sql
SELECT category.cat_name, product.prod_name FROM category
JOIN product ON product.prod_cat = category.cat_id
ORDER BY category.cat_name, product.prod_id;
```

LANGUAGE: Unknown

```
  cat_name    |                     prod_name
--------------+-------------------------------------------------
 Health       | Balanced client-server product
 Health       | Pre-emptive holistic intranet
 Health       | Multi-tiered explicit paradigm
 Health       | Monitored asynchronous function
 Health       | Right-sized mission-critical pricing structure
 Health       | Open-architected homogeneous concept
 Health       | Fully-configurable full-range interface
 Health       | Customizable cohesive capacity
 Health       | Seamless optimal leverage
 Health       | Realigned client-driven database
 Health       | Profound human-resource forecast
 Health       | User-friendly encompassing array
 Health       | Networked global open system
 Health       | Team-oriented stable project
 Health       | Exclusive multimedia middleware
 Health       | Ameliorated next generation orchestration
 Kid's Wear   | Cross-platform fresh-thinking core
 Kid's Wear   | Profound optimal encryption
 Kid's Wear   | Business-focused holistic help-desk
 Kid's Wear  | Configurable analyzing solution
 Kid's Wear  | Monitored non-volatile initiative
 Kid's Wear  | Pre-emptive next generation infrastructure
 Kid's Wear  | Persevering empowering customer loyalty
 Kid's Wear  | Progressive modular archive
 Kid's Wear  | Cross-group reciprocal firmware
 Kid's Wear  | Advanced neutral portal
 Kid's Wear  | Customer-focused needs-based protocol
 Kid's Wear  | Fundamental global archive
 Kid's Wear  | Digitized tertiary groupware
 Kid's Wear  | Total intangible artificial intelligence
 Kid's Wear  | Organic clear-thinking system engine
 Kid's Wear  | Persistent incremental model
 Ladies Wear | Multi-layered multi-tasking initiative
 Ladies Wear | Robust directional projection
 Ladies Wear | Re-engineered explicit software
 Ladies Wear | Multi-channelled well-modulated analyzer
 Ladies Wear | Public-key interactive encoding
 Ladies Wear | Realigned 5th generation artificial intelligence
 Ladies Wear | Vision-oriented user-facing framework
 Ladies Wear | Secured holistic hierarchy
 Ladies Wear | Assimilated regional instruction set
 Ladies Wear | Virtual impactful success
 Ladies Wear | Universal global hub
 Ladies Wear | Adaptive modular approach
 Ladies Wear | Synergistic zero defect info-mediaries
 Ladies Wear | Reduced fresh-thinking process improvement
 Ladies Wear | Stand-alone composite Graphical User Interface
 Ladies Wear | Decentralized human-resource infrastructure
 Ladies Wear | Balanced modular website
 Ladies Wear | Horizontal explicit benchmark
 Ladies Wear | Innovative web-enabled extranet
 Ladies Wear | Exclusive analyzing open architecture
 Ladies Wear | Integrated 24/7 interface
 Ladies Wear | Ergonomic solution-oriented local area network
 Men's Wear  | Operative analyzing task-force
 Men's Wear  | Re-engineered cohesive methodology
 Men's Wear  | Balanced real-time info-mediaries
 Men's Wear  | Implemented optimizing benchmark
 Men's Wear  | Adaptive static website
 Men's Wear  | Diverse reciprocal knowledge base
 Men's Wear  | Robust foreground leverage
 Men's Wear  | Re-engineered 24/7 knowledge base
 Men's Wear  | Advanced didactic Graphic Interface
 Outdoor     | Inverse transitional infrastructure
 Outdoor     | Diverse neutral emulation
 Outdoor     | Up-sized composite challenge
 Outdoor     | Intuitive directional complexity
 Outdoor     | Re-engineered actuating capability
 Outdoor     | Proactive methodical data-warehouse
 Outdoor     | Switchable tangible product
 Outdoor     | Enhanced discrete function
 Outdoor     | Horizontal asynchronous intranet
 Outdoor     | Switchable 5th generation parallelism
 Outdoor     | 4th generation Graphical User Interface
 Outdoor     | Future-proofed leading edge customer loyalty
 Outdoor     | Enhanced homogeneous paradigm
 Outdoor     | Inverse high-level attitude
 Outdoor     | Quality-focused upward-trending throughput
 Sport       | Exclusive client-server array
 Sport       | Exclusive background website
 Sport       | Universal encompassing conglomeration
 Sport       | Synergistic homogeneous ability
 Sport       | Open-source impactful archive
 Sport       | Configurable methodical firmware
 Sport       | Virtual stable Graphic Interface
 Sport       | Realigned homogeneous hub
 Sport       | Quality-focused foreground analyzer
 Sport       | Universal exuding protocol
 Sport       | Balanced hybrid portal
 Sport       | Customizable well-modulated encryption
 Sport       | Business-focused solution-oriented moratorium
 Sport       | Synergistic scalable capability
 Sport       | Distributed uniform Graphic Interface
 Sport       | Profound value-added intranet
 Sport       | Persistent demand-driven complexity
 Sport       | Focused secondary initiative
 Sport       | Streamlined asynchronous functionalities
 Sport       | Vision-oriented attitude-oriented core
 Sport       | Front-line demand-driven utilisation
 Sport       | Robust mission-critical complexity
(100 rows)
```

6. How can you prove that the product id is being used to do the ordering? (You may have already done this in Q5). Copy the code and the answer below.

LANGUAGE: SQL

```sql
SELECT category.cat_name, product.prod_name, product.prod_id FROM category
JOIN product ON product.prod_cat = category.cat_id
ORDER BY category.cat_name, product.prod_id;
```

LANGUAGE: Unknown

```
  cat_name   |                    prod_name                     | prod_id
-------------+--------------------------------------------------+---------
 Health      | Balanced client-server product                   |       4
 Health      | Pre-emptive holistic intranet                    |       6
 Health      | Multi-tiered explicit paradigm                   |      10
 Health      | Monitored asynchronous function                  |      20
 Health      | Right-sized mission-critical pricing structure   |      23
 Health      | Open-architected homogeneous concept             |      37
 Health      | Fully-configurable full-range interface          |      46
 Health      | Customizable cohesive capacity                   |      54
 Health      | Seamless optimal leverage                        |      57
 Health      | Realigned client-driven database                 |      59
 Health      | Profound human-resource forecast                 |      69
 Health      | User-friendly encompassing array                 |      72
 Health      | Networked global open system                     |      81
 Health      | Team-oriented stable project                     |      88
 Health      | Exclusive multimedia middleware                  |      94
 Health      | Ameliorated next generation orchestration        |      95
 Kid's Wear  | Cross-platform fresh-thinking core               |      12
 Kid's Wear  | Profound optimal encryption                      |      28
 Kid's Wear  | Business-focused holistic help-desk              |      32
 Kid's Wear  | Configurable analyzing solution                  |      45
 Kid's Wear  | Monitored non-volatile initiative                |      47
 Kid's Wear  | Pre-emptive next generation infrastructure       |      48
 Kid's Wear  | Persevering empowering customer loyalty          |      52
 Kid's Wear  | Progressive modular archive                      |      55
 Kid's Wear  | Cross-group reciprocal firmware                  |      62
 Kid's Wear  | Advanced neutral portal                          |      70
 Kid's Wear  | Customer-focused needs-based protocol            |      71
 Kid's Wear  | Fundamental global archive                       |      79
 Kid's Wear  | Digitized tertiary groupware                     |      84
 Kid's Wear  | Total intangible artificial intelligence         |      89
 Kid's Wear  | Organic clear-thinking system engine             |      97
 Kid's Wear  | Persistent incremental model                     |      98
 Ladies Wear | Multi-layered multi-tasking initiative           |       1
 Ladies Wear | Robust directional projection                    |       8
 Ladies Wear | Re-engineered explicit software                  |      11
 Ladies Wear | Multi-channelled well-modulated analyzer         |      17
 Ladies Wear | Public-key interactive encoding                  |      19
 Ladies Wear | Realigned 5th generation artificial intelligence |      26
 Ladies Wear | Vision-oriented user-facing framework            |      29
 Ladies Wear | Secured holistic hierarchy                       |      30
 Ladies Wear | Assimilated regional instruction set             |      31
 Ladies Wear | Virtual impactful success                        |      36
 Ladies Wear | Universal global hub                             |      41
 Ladies Wear | Adaptive modular approach                        |      50
 Ladies Wear | Synergistic zero defect info-mediaries           |      51
 Ladies Wear | Reduced fresh-thinking process improvement       |      56
 Ladies Wear | Stand-alone composite Graphical User Interface   |      67
 Ladies Wear | Decentralized human-resource infrastructure      |      73
 Ladies Wear | Balanced modular website                         |      74
 Ladies Wear | Horizontal explicit benchmark                    |      75
 Ladies Wear | Innovative web-enabled extranet                    |      77
 Ladies Wear | Exclusive analyzing open architecture              |      78
 Ladies Wear | Integrated 24/7 interface                          |      92
 Ladies Wear | Ergonomic solution-oriented local area network     |      99
 Men's Wear  | Operative analyzing task-force                     |       2
 Men's Wear  | Re-engineered cohesive methodology                 |       7
 Men's Wear  | Balanced real-time info-mediaries                  |      22
 Men's Wear  | Implemented optimizing benchmark                   |      34
 Men's Wear  | Adaptive static website                            |      35
 Men's Wear  | Diverse reciprocal knowledge base                  |      38
 Men's Wear  | Robust foreground leverage                         |      53
 Men's Wear  | Re-engineered 24/7 knowledge base                  |      76
 Men's Wear  | Advanced didactic Graphic Interface                |      93
 Outdoor     | Inverse transitional infrastructure                |       9
 Outdoor     | Diverse neutral emulation                          |      13
 Outdoor     | Up-sized composite challenge                       |      14
 Outdoor     | Intuitive directional complexity                   |      15
 Outdoor     | Re-engineered actuating capability                 |      18
 Outdoor     | Proactive methodical data-warehouse                |      21
 Outdoor     | Switchable tangible product                        |      40
 Outdoor     | Enhanced discrete function                         |      42
 Outdoor     | Horizontal asynchronous intranet                   |      43
 Outdoor     | Switchable 5th generation parallelism              |      49
 Outdoor     | 4th generation Graphical User Interface            |      63
 Outdoor     | Future-proofed leading edge customer loyalty       |      68
 Outdoor     | Enhanced homogeneous paradigm                      |      85
 Outdoor     | Inverse high-level attitude                        |      86
 Outdoor     | Quality-focused upward-trending throughput         |      87
 Sport       | Exclusive client-server array                      |       3
 Sport       | Exclusive background website                       |       5
 Sport       | Universal encompassing conglomeration              |      16
 Sport       | Synergistic homogeneous ability                    |      24
 Sport       | Open-source impactful archive                      |      25
 Sport       | Configurable methodical firmware                   |      27
 Sport       | Virtual stable Graphic Interface                   |      33
 Sport       | Realigned homogeneous hub                          |      39
 Sport       | Quality-focused foreground analyzer                |      44
 Sport       | Universal exuding protocol                         |      58
 Sport       | Balanced hybrid portal                             |      60
 Sport       | Customizable well-modulated encryption             |      61
 Sport       | Business-focused solution-oriented moratorium      |      64
 Sport       | Synergistic scalable capability                    |      65
 Sport       | Distributed uniform Graphic Interface              |      66
 Sport       | Profound value-added intranet                      |      80
 Sport       | Persistent demand-driven complexity                |      82
 Sport       | Focused secondary initiative                       |      83
 Sport       | Streamlined asynchronous functionalities           |      90
 Sport       | Vision-oriented attitude-oriented core             |      91
 Sport       | Front-line demand-driven utilisation               |      96
 Sport       | Robust mission-critical complexity                 |     100
(100 rows)
```

7. Write a query that will list all staff members first and last names along with their email addresses that are cleaners. Copy the code and the answer below.

LANGUAGE: SQL

```sql
SELECT staff.staff_fname, staff.staff_lname, staff.work_email FROM staff
JOIN role ON staff.role=role.role_id
WHERE role.role_name='Cleaner';
```

LANGUAGE: Unknown

```
 staff_fname | staff_lname | work_email
-------------+-------------+------------
(0 rows)
```

8. How many staff are there who have the role Misc? Copy the code and the answer below.

LANGUAGE: SQL

```sql
SELECT count(*) FROM staff
JOIN role ON staff.role = role.role_id
WHERE role.role_name='Misc';
```

LANGUAGE: Unknown

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

9. What are the addresses of the staff that are returned by the query for Q8? You should output their first and last names too. Copy the code and the answer below.

LANGUAGE: SQL

```sql
SELECT staff.staff_fname, staff.staff_lname, concat_ws(' ', addr1, addr2, town, postcode) AS "
    ↪ address"
FROM staff
JOIN role ON role.role_id = staff.role
WHERE role.role_name='Misc';
```

LANGUAGE: Unknown

```
 staff_fname | staff_lname |                    address
-------------+-------------+-----------------------------------------------
 Janeva      | Gillicuddy  | 6999 Kings Park Sachtjen Portsmouth PO05 5SF
 Nell        | Olsson      | 18424 Kenwood Court Farmco Havant PO22 6DL
 Tim         | Illem       | 85 Lillian Way Farragut Southsea PO93 0CN
(3 rows)
```

10. List the product id numbers with their names that start with the letters Re . Copy the code and the answer below.

LANGUAGE: SQL

```sql
SELECT prod_id, prod_name FROM product
WHERE prod_name LIKE 'Re%';
```

LANGUAGE: Unknown

```
 prod_id |                     prod_name
---------+---------------------------------------------------
       7 | Re-engineered cohesive methodology
      11 | Re-engineered explicit software
      18 | Re-engineered actuating capability
      26 | Realigned 5th generation artificial intelligence
      39 | Realigned homogeneous hub
      56 | Reduced fresh-thinking process improvement
      59 | Realigned client-driven database
      76 | Re-engineered 24/7 knowledge base
(8 rows)
```

11. List the product id numbers with their names that have the word `value` in the name somewhere. Copy the code and the answer below.

LANGUAGE: SQL

```sql
SELECT prod_id, prod_name FROM product
WHERE prod_name LIKE '%value%';
```

LANGUAGE: Unknown

```
 prod_id |           prod_name
---------+------------------------------
      80 | Profound value-added intranet
(1 row)
```

12. List the product names along with their id numbers that have `Value` somewhere in their name. Copy the code and the answer below

LANGUAGE: SQL

```sql
SELECT prod_id, prod_name FROM product
WHERE prod_name LIKE '%Value%';
```

LANGUAGE: Unknown

```
 prod_id | prod_name
---------+-----------
(0 rows)
```

13. List the customer first and last names along with their email addresses, the customer order id, the category names and the product names for orders that have been placed for all products that have the word `able` in the name. (The case matters). Order by the category and the product name. The output should have the category names in alphabetical order then within each category the products should be ordered in alphabetical order. Copy the code and the answer below.

LANGUAGE: SQL

```sql
SELECT customer.cust_fname, customer.cust_lname, customer.email, cust_order.cust_ord_id,
    ↪ category.cat_name, product.prod_name from customer
JOIN cust_order ON customer.cust_id=cust_order.cust_id
JOIN manifest ON cust_order.cust_ord_id=manifest.cust_ord_id
JOIN product on manifest.prod_id=product.prod_id
JOIN category on category.cat_id=product.prod_cat
WHERE product.prod_name LIKE '%able%'
ORDER BY category.cat_name, product.prod_name;
```

LANGUAGE: Unknown

```
   cust_fname     |    cust_lname     |               email                | cust_ord_id |  cat_name
    ↪    |
 prod_name
-----------------+------------------+----------------------------------+-------------+------------+------------
    ↪
 Bérengère        | Menendez          | amenendez3@dell.com                |          64 | Health
    ↪    | Customizable cohesive capacity
 Marie-françoise | Currier           | acurrier0@economist.com            |         133 | Health
    ↪    | Customizable cohesive capacity
 Bérengère        | Menendez          | amenendez3@dell.com                |         102 | Health
    ↪    | Fully-configurable full-range interface
 Chadd            | Franz-Schoninger | cfranzschoninger3@google.com.hk |           7 | Health
    ↪    | Team-oriented stable project
 Chadd            | Franz-Schoninger | cfranzschoninger3@google.com.hk |          81 | Health
    ↪    | Team-oriented stable project
 Bénédicte        | Dozdill           | cdozdill1@amazon.de                |          24 | Kid's
    ↪ Wear | Configurable analyzing solution
 Bérengère        | Menendez          | amenendez3@dell.com                |          21 | Kid's
    ↪ Wear | Configurable analyzing solution
 Bérengère        | Menendez          | amenendez3@dell.com                |         113 | Kid's
    ↪ Wear | Configurable analyzing solution
 Jobey            | Boeter            | jboeter0@mail.ru                   |          91 | Kid's
    ↪ Wear | Configurable analyzing solution
 Jobey            | Boeter            | jboeter0@mail.ru                   |          39 | Outdoor
```

```
     ↪    | Switchable tangible product
 Jobey           | Boeter           | jboeter0@mail.ru                  |          26 | Outdoor
     ↪    | Switchable tangible product
 Vikky           | Eke              | veke4@elegantthemes.com           |         105 | Sport
     ↪    | Configurable methodical firmware
 Vikky           | Eke              | veke4@elegantthemes.com           |         118 | Sport
     ↪    | Customizable well-modulated encryption
 Pélagie         | Hachard          | fhachard4@blinklist.com           |          89 | Sport
     ↪    | Virtual stable Graphic Interface
(14 rows)
```

S.3. SECURITY BASICS

08-12-22 13:00 Mark RB LT1

This lecture has been split into two parts, the second part will take place after the Christmas break.
Next week's lecture will be part about MS Learn (& part about Databases) and the practical next week is optional, aimed around coursework questions.

A View on Security

Stealing data is very different to stealing physical objects. To steal data, you just have to make a copy of it; whereas with physical things, you have to pick up the physical thing.
At one time, physical security was talked about much more. Nowerdays, the physical hardware is stored on the cloud where this is dealt with by someone else.
When working on developing applications, you have to 'sanitise' data which is passed to the database.
The biggest risk to data is those who have access to it, generally this will be people who work for the company.

PostgreSQL Basic Security

Our user account in our Postgres install has full administrative rights to Postgres. This is the Superuser account which no one else should have access to. By default, you cannot access the server from a different IP address; it is possible to allow other IP addresses to have access to this however this is un-advised.
Currently, the superuser on our databases doesn't have a password. In the real world, this is very stupid and should never happen. As superusers we can change and set other users passwords.

Roles

In Postgres, a role is the same as a user.
Before you can login to Postgres, there has to be a role in the DBMS to allow you to login. This username is case sensitive.
As well as having a role/ user there has to be other things in the database. For us, this is the table called our up number.
Users should (in the real world, must) be given passwords. Constraints and change-after-time policies can be set. When the user is created, the password is set. This is a potential security risk as if someone else can get into your account, they can view your terminal history, including the passwords you've entered in terminal in plain text.
Users have to be given the ability to log in. Removing the log in ability, can be useful for people who are working temporarily for a company.
The syntax to create a role as follows:

LANGUAGE: SQL
```sql
CREATE role [userName] with login password '[password]';
```

Where `[userName]` and `[password]` are replaced with values you wish to enter.
There is also a `CREATE user` command however this returns the same value as `CREATE role`.

When creating a role, this will create a database called their username, this is essential and should not be deleted.
After creating a role, you have to specify permissions for the different users. However, you can login (if you have login permission) and see all the names of all the databases.

Views

Including views in the coursework will give additional marks.

View

A pre-written query

This enables us to delegate access to certain parts of a table.
When you create views, you can give users access to be able to run that query.
To create a view, the syntax follows

LANGUAGE: SQL

```sql
CREATE [viewName] AS [queryString];

--eg
CREATE VIEW CUST_NAMES AS SELECT CUST_FNAME, CUST_LNAME FROM customer;
```

The view above can be executed as

LANGUAGE: SQL

```sql
SELECT * FROM CUST_NAMES;
```

This will display a list of all the customers first names and customers last names.

S.4. PRACTICAL: MORE JOINS

08-12-22 14:00 Mark & Co FTC 3

1. Once you have run the code in this week's tutor section, write a left join that joins the customer and cust_order tables.

LANGUAGE: SQL

```sql
SELECT customer.cust_fname, customer.cust_lname, cust_order.cust_ord_id FROM customer
LEFT JOIN cust_order ON customer.cust_id=cust_order.cust_id;
```

LANGUAGE: Unknown

```
   cust_fname     |    cust_lname     | cust_ord_id
------------------+-------------------+-------------
 Chadd            | Franz-Schoninger  |           1
 York             | O'Deegan          |           2
 Marie-françoise  | Currier           |           3
 Bérengère        | Menendez          |           4
 Bénédicte        | Dozdill           |           5
 Bénédicte        | Dozdill           |           6
 Chadd            | Franz-Schoninger  |           7
 Bénédicte        | Dozdill           |           8
 Penelope         | Hexter            |           9
 York             | O'Deegan          |          10
 ...
(252 rows)
```

2. Write a right join that joins the customer and cust_order tables

LANGUAGE: SQL

```sql
SELECT customer.cust_fname, customer.cust_lname, cust_order.cust_ord_id FROM customer
RIGHT JOIN cust_order ON customer.cust_id=cust_order.cust_id;
```

LANGUAGE: Unknown

```
   cust_fname     |    cust_lname     | cust_ord_id
------------------+-------------------+-------------
 Chadd            | Franz-Schoninger  |           1
 York             | O'Deegan          |           2
 Marie-françoise  | Currier           |           3
 Bérengère        | Menendez          |           4
 Bénédicte        | Dozdill           |           5
 Bénédicte        | Dozdill           |           6
 Chadd            | Franz-Schoninger  |           7
 Bénédicte        | Dozdill           |           8
 Penelope         | Hexter            |           9
 York             | O'Deegan          |          10
 Bénédicte        | Dozdill           |          11
 ...
(250 rows)
```

3. write an inner join that joins the customer and cust_order tables.

LANGUAGE: SQL

```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;
```

LANGUAGE: Unknown

```
   cust_fname     |    cust_lname     | cust_ord_id
------------------+-------------------+-------------
 Chadd            | Franz-Schoninger  |           1
 York             | O'Deegan          |           2
 Marie-françoise  | Currier           |           3
 Bérengère        | Menendez          |           4
 Bénédicte        | Dozdill           |           5
 Bénédicte        | Dozdill           |           6
 Chadd            | Franz-Schoninger  |           7
 Bénédicte        | Dozdill           |           8
 Penelope         | Hexter            |           9
 York             | O'Deegan          |          10
 ...
(250 rows)
```

4. Write a right join that joins the customer and cust_order tables.

LANGUAGE: SQL

```sql
SELECT customer.cust_fname, customer.cust_lname, cust_order.cust_ord_id FROM customer
RIGHT JOIN cust_order ON customer.cust_id=cust_order.cust_id;
```

LANGUAGE: Unknown

```
   cust_fname     |    cust_lname     | cust_ord_id
------------------+-------------------+-------------
 Chadd            | Franz-Schoninger  |           1
 York             | O'Deegan          |           2
 Marie-françoise  | Currier           |           3
 Bérengère        | Menendez          |           4
 Bénédicte        | Dozdill           |           5
 Bénédicte        | Dozdill           |           6
 Chadd            | Franz-Schoninger  |           7
 Bénédicte        | Dozdill           |           8
 Penelope         | Hexter            |           9
 York             | O'Deegan          |          10
 ...
(251 rows)
```

5. Write an inner join that joins the customer and cust_order tables.

LANGUAGE: SQL

```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;
```

LANGUAGE: Unknown

```
   cust_fname     |    cust_lname     | cust_ord_id
------------------+-------------------+-------------
 Chadd            | Franz-Schoninger  |           1
 York             | O'Deegan          |           2
 Marie-françoise  | Currier           |           3
 Bérengère        | Menendez          |           4
 Bénédicte        | Dozdill           |           5
 Bénédicte        | Dozdill           |           6
 Chadd            | Franz-Schoninger  |           7
 Bénédicte        | Dozdill           |           8
 Penelope         | Hexter            |           9
 York             | O'Deegan          |          10
 ...
(251 rows)
```

6. Write a left join that joins the customer and cust_order tables.

LANGUAGE: SQL

```sql
SELECT customer.cust_fname, customer.cust_lname, cust_order.cust_ord_id FROM customer
LEFT JOIN cust_order ON customer.cust_id=cust_order.cust_id;
```

LANGUAGE: Unknown

```
   cust_fname     |    cust_lname     | cust_ord_id
------------------+-------------------+-------------
 Chadd            | Franz-Schoninger  |           1
 York             | O'Deegan          |           2
 Marie-françoise  | Currier           |           3
 Bérengère        | Menendez          |           4
 Bénédicte        | Dozdill           |           5
 Bénédicte        | Dozdill           |           6
 Chadd            | Franz-Schoninger  |           7
 Bénédicte        | Dozdill           |           8
 Penelope         | Hexter            |           9
 York             | O'Deegan          |          10
 ...
(262 rows)
```

7. Rewrite the query for number 6 but reverse the order of the tables. If you started with the customer table in the query and joined cust_order then rewrite starting with cust_order and join customer.

LANGUAGE: SQL

```sql
SELECT customer.cust_fname, customer.cust_lname, cust_order.cust_ord_id FROM cust_order
LEFT JOIN customer ON customer.cust_id=cust_order.cust_id;
```

LANGUAGE: Unknown

```
   cust_fname     |    cust_lname     | cust_ord_id
------------------+-------------------+-------------
 Chadd            | Franz-Schoninger  |           1
 York             | O'Deegan          |           2
 Marie-françoise  | Currier           |           3
 Bérengère        | Menendez          |           4
 Bénédicte        | Dozdill           |           5
 Bénédicte        | Dozdill           |           6
 Chadd            | Franz-Schoninger  |           7
 Bénédicte        | Dozdill           |           8
 Penelope         | Hexter            |           9
 York             | O'Deegan          |          10
 ...
(251 rows)
```

8. Depending on the number of rows that are returned from questions 6 and 7, rewrite the one that has the highest number of results so that the result is sorted firstly by the cust_id and then the cust_ord_id. Copy the query AND THE FIRST SCREEN OF DATA RETURNED BELOW. Make sure you have more than 1 cust_id in the results.

LANGUAGE: SQL

```sql
-- use query from question 6
SELECT customer.cust_fname, customer.cust_lname, cust_order.cust_ord_id FROM cust_order
LEFT JOIN customer ON customer.cust_id=cust_order.cust_id
ORDER BY customer.cust_id, cust_order.cust_ord_id;
```

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
 York            | O'Deegan         |           2
 York            | O'Deegan         |          10
 York            | O'Deegan         |          19
 ...
(251 rows)
```

9. Write a query that uses outer joins on the customer, the cust_order table and the staff table. It must return the cust_id, cust_ord_id and the staff_id as well as the staff members last name and their work email address.

LANGUAGE: SQL

```sql
SELECT c.cust_id, co.cust_ord_id, s.staff_id, s.staff_lname, s.work_email FROM customer c
FULL OUTER JOIN cust_order co ON c.cust_id=co.cust_id
FULL OUTER JOIN staff s ON s.staff_id=co.staff_id;
```

LANGUAGE: Unknown

```
 cust_id | cust_ord_id | staff_id | staff_lname |         work_email
---------+-------------+----------+-------------+----------------------------
       4 |           1 |        6 | Clewlowe    | Aura.Clewlowe@dsd.com
       2 |           2 |        5 | Gillicuddy  | Janeva.Gillicuddy@dsd.com
       6 |           3 |        2 | Shrimpton   | Nikoletta.Shrimpton@dsd.com
       9 |           4 |        5 | Gillicuddy  | Janeva.Gillicuddy@dsd.com
       7 |           5 |        6 | Clewlowe    | Aura.Clewlowe@dsd.com
       7 |           6 |        4 | Gloster     | Hanan.Gloster@dsd.com
       4 |           7 |        6 | Clewlowe    | Aura.Clewlowe@dsd.com
       7 |           8 |        3 | Housegoe    | Montgomery.Housegoe@dsd.com
       3 |           9 |        6 | Clewlowe    | Aura.Clewlowe@dsd.com
       2 |          10 |        5 | Gillicuddy  | Janeva.Gillicuddy@dsd.com
       7 |          11 |        6 | Clewlowe    | Aura.Clewlowe@dsd.com
       9 |          12 |        4 | Gloster     | Hanan.Gloster@dsd.com
       7 |          13 |        4 | Gloster     | Hanan.Gloster@dsd.com
       7 |          14 |        4 | Gloster     | Hanan.Gloster@dsd.com
       6 |          15 |        4 | Gloster     | Hanan.Gloster@dsd.com
       9 |          16 |        5 | Gillicuddy  | Janeva.Gillicuddy@dsd.com
      10 |          17 |        5 | Gillicuddy  | Janeva.Gillicuddy@dsd.com
       7 |          18 |        3 | Housegoe    | Montgomery.Housegoe@dsd.com
       2 |          19 |        3 | Housegoe    | Montgomery.Housegoe@dsd.com
...
(266 rows)
```

10. Rewrite the query from 9 and filter the results to show only those customers who have not placed an order. (Remember that any customer who has placed an order will have a cust_ord_id associated with them).

LANGUAGE: SQL

```sql
SELECT c.cust_id, co.cust_ord_id, s.staff_id, s.staff_lname, s.work_email FROM customer c
FULL OUTER JOIN cust_order co ON c.cust_id=co.cust_id
FULL OUTER JOIN staff s ON s.staff_id=co.staff_id
WHERE co.cust_ord_id IS NULL AND c.cust_id IS NOT NULL;
```

LANGUAGE: Unknown

```
 cust_id | cust_ord_id | staff_id | staff_lname | work_email
---------+-------------+----------+-------------+------------
      25 |             |          |             |
      27 |             |          |             |
      33 |             |          |             |
      31 |             |          |             |
      34 |             |          |             |
      32 |             |          |             |
      24 |             |          |             |
      28 |             |          |             |
      30 |             |          |             |
      29 |             |          |             |
      35 |             |          |             |
(11 rows)
```

11. Write a query that will display the staff first and last names, their work email addresses, the customer order id, the customer id and the customer's first and last names along with the products that are in the customer's orders. The results must be ordered by customer last name order. Copy the query AND THE FIRST SCREEN OF DATA RETURNED BELOW. (Make sure you have more than 1 customer in the results).

LANGUAGE: SQL

```sql
SELECT s.staff_fname, s.staff_lname, s.work_email, co.cust_ord_id, c.cust_id, c.cust_fname, c.
    ↪ cust_lname, p.prod_name FROM customer c
JOIN cust_order co ON c.cust_id=co.cust_id
JOIN staff s ON s.staff_id=co.staff_id
JOIN manifest ON manifest.cust_ord_id = co.cust_ord_id
JOIN product p ON p.prod_id = manifest.prod_id
ORDER BY c.cust_lname;
```

LANGUAGE: Unknown

```
 staff_fname | staff_lname |         work_email          | cust_ord_id | cust_id |    cust_fname
    ↪      |    cust_lname     |                    prod_name
-------------+-------------+-----------------------------+-------------+---------+------------------+----------
    ↪
 Hanan       | Gloster     | Hanan.Gloster@dsd.com       |          39 |       1 | Jobey
    ↪      | Boeter            | Switchable tangible product
 Nikoletta   | Shrimpton   | Nikoletta.Shrimpton@dsd.com |          57 |       1 | Jobey
    ↪      | Boeter            | Persistent demand-driven complexity
 Montgomery  | Housegoe    | Montgomery.Housegoe@dsd.com |          68 |       1 | Jobey
    ↪      | Boeter            | Streamlined asynchronous functionalities
 Aura        | Clewlowe    | Aura.Clewlowe@dsd.com       |         131 |       1 | Jobey
    ↪      | Boeter            | Seamless optimal leverage
 Janeva      | Gillicuddy  | Janeva.Gillicuddy@dsd.com   |          99 |       1 | Jobey
    ↪      | Boeter            | Fundamental global archive
 Hanan       | Gloster     | Hanan.Gloster@dsd.com       |          34 |       1 | Jobey
    ↪      | Boeter            | Right-sized mission-critical pricing structure
 Montgomery  | Housegoe    | Montgomery.Housegoe@dsd.com |          26 |       1 | Jobey
    ↪      | Boeter            | Switchable tangible product
 Hanan       | Gloster     | Hanan.Gloster@dsd.com       |          77 |       1 | Jobey
    ↪      | Boeter            | Realigned homogeneous hub
 Montgomery  | Housegoe    | Montgomery.Housegoe@dsd.com |         146 |       1 | Jobey
    ↪      | Boeter            | Fundamental global archive
 Janeva      | Gillicuddy  | Janeva.Gillicuddy@dsd.com   |         143 |       1 | Jobey
    ↪      | Boeter            | Re-engineered cohesive methodology
 Niel        | Welsby      | Niel.Welsby@dsd.com         |          91 |       1 | Jobey
    ↪      | Boeter            | Configurable analyzing solution
 Nikoletta   | Shrimpton   | Nikoletta.Shrimpton@dsd.com |          71 |       1 | Jobey
    ↪      | Boeter            | Inverse high-level attitude
 Montgomery  | Housegoe    | Montgomery.Housegoe@dsd.com |          98 |       1 | Jobey
    ↪      | Boeter            | Distributed uniform Graphic Interface
 Niel        | Welsby      | Niel.Welsby@dsd.com         |         112 |       6 | Marie-
    ↪ françoise | Currier           | Integrated 24/7 interface
...
(150 rows)
```

12. Write a query that will show only the customer contact details who have NEVER placed an order. It is up to you to decide what we mean by contact details. Copy the output and query below.

LANGUAGE: SQL

```sql
SELECT c.cust_fname, c.email FROM customer c
FULL OUTER JOIN cust_order co ON c.cust_id = co.cust_id
WHERE co.cust_ord_id IS NULL;
```

LANGUAGE: Unknown

```
 cust_fname |               email
------------+-----------------------------------
 Jen        | jsettle222@google.ca
 Fawnia     | fpetchell1@networkadvertising.org
 Nealy      | nstanley7@arstechnica.com
 Tine       | tclopton5@typepad.com
 Cody       | clago8@rambler.ru
 Lonnie     | lmacgilpatrick6@uiuc.edu
 Evie       | 3vi3@google.wh
 Mireielle  | mkillner2@cafepress.com
 Falkner    | fgrouer4@dion.ne.jp
 Kaine      | klawford3@imdb.com
 Theadora   | tajsik9@sfgate.com
(11 rows)
```