Databases and SQL:

DBMS: Database Management Systems.

* Such as: Mongo, SQL etc.
* DB’s are methods of keeping records.
* Fields = Items of information stored in a DB.
* Collection of fields = record
* A table is a collection of records = a folder
* A collection of tables = a Database
* No field = no record
* The field is the physical thing that exists
* If there is a field = there is a record
* You need to make the structure of the record my friend
* The record is the information in the structure of the record.
* You need to start by making the structure of the collection of the fields
* You need at least three factors of information to make a usable field (?)
* **Name**, **capacity** (size of the field in number of characters), and **what type of data file you are storing** – numeric (the data that allows us to use it for mathematical calculations), and non-numeric.
* My friend

We will be using SQL language:

* Industry standard for databases
* Structured Query Language
* Collection of commands used to communicate with the DBMS
* 15-20 commands
* Split into groups:
* DML: Data Manipulation Language

Commands: Insert (insert a record in the structure. Cannot be done without a structure). Update (makes change in the record). Delete (remove the record).

* DDL: Data Definition Language

Commands: Create (the command to create the structure). Alter (edits the structure). Drop (removes the structure altogether). Follow these with the structure you want to edit. The changes you can make to a structure.

* DRL: Data Retrieval Language

Command: Select (shows the specified data from the table)

* DCL: Data Control Language

Commands: Grant (gives rights to user) Revoke (removes rights from user)

* Transaction Control

Commands: Rollback, Commit, and SavePoint

Oracle:

* Username: HR
* Password: oracle

Making a table, school example:

* Create Table School,
* Reg Number Number(2),
* Name char(30),
* Marks Number (3)

) ; Insert the records:

* > insert into school values (25, ‘Shafeeq’, 75); -Press ENTER-
* 1st value in the brackets will go to the 1st column, 2nd to the 2nd and so on.
* > insert into school values (36, ‘Peter’, 82); -Press ENTER-
* > select \* from school; -ENTER- (asterisk shows every value in the table)

This process isn’t case sensitive.

Shop table:

* Create table Tesco (Product Char(20), Quantity Number(2), Price Number (3, 2)
* Describe Command: Describe Tesco.
* Will show you the structure of your table, not the records in the table.
* Insert into Tesco values (‘Seven Up’, 15, 1.25)
* Insert into Tesco values (‘Coke’, 10, 2.0)
* Select \* from Tesco
* Select Product, quantity \* price from Tesco
* To change heading type Select Product as Yadda Yadda Yadda
* ***Syntax doesn’t like spaces***
* Double quotes (“”) is for headings
* To add a products VAT:
* Select Product , Quantity\*Price As Amount/100AsVATfromTesco
* Select Product, Quantity\*Price As Amount, (Quantity\*Price)\*15/100 as VAT, (Quantity\*Price) + (Quantity\*Price) as Bill From Tesco

Relation operators:

* <
* >
* Columns, constants, expressions can be compared with another.

In the schools example:

* Select \* from schools where 3 > 2

Logical operators:

* And (when used all conditions must be true/false for the statement to pass)
* Or (when used one of the conditions must be true/false to get true/false)
* Not = Boolean operator

And/or are binary. All logical operators work on Boolean values (true and false).

And vs. or

* True and True = True
* True and False = False
* False and True = False
* False and False = False
* True or True = True
* True or False = True
* False or True = True
* False or False = False

Select distinct \* from [value] – shows every instance of a certain value in the table.

* Select \* from column where value = [value]
* If [value] is found = true, if not, = false.

To arrange in a specific order, in Tesco example:

* Select \* from Tesco order by Price
* To arrange by Descending: add “Desc” at the end of the query
* Values will be arranged ascending by default.

Every corrector has an ASCII code (American Standard Code for Information Interchange)

* A = 65
* B = 66
* A = 97
* B = 98
* 0 = 78
* 1 = 79
* SPACE = 32
* CPU will sort ASCII code in ascending order my friend

If there is a duplication in any values the query is asking to sort on multiple factors it will sort them by the first factor.

To change a record in the table, tesco example:

* Update tesco set Price = ‘1.20’ where Product = ‘7Up’
* Include single quotes of names/values

To delete a record, tesco example:

* Delete from tesco (will delete all records)
* Delete from tesco where Product = Fanta
* Delete from tesco where Price > 1.0

To insert a record when you don’t have all the values, tesco example:

* Insert into tesco (Product, Price) values (‘Relentless’, ‘1.85’)
* To insert the remaining value: update school set Quantity ‘15’ where Product = Relentless (for example)

Functions:

* Group functions: The function works on the group of values
* = Functions such as sum, max, min, count (just counts the number of records), avg (counts the average of the values).

Functions applied in Tesco example:

* Select sum(Quantity) from tesco – adds the quantity column together to work out total quantity.
* For max: select max(Price) from tesco – shows max Price from table
* And so on…

To find the maximum quantity in the tesco example:

* Select \* from tesco where quantity = (select max(quantity) from tesco)
* Outer query (left) = Sub query (right)
* Another example: select \* from tesco where Price <(select avg(Price)

Tesco example, find the maximum Quantity where the Price is 1.85.

* Select \* from tesco where Quantity = (select max(Quantity) from tesco where Price = ‘1.85’) and Price = ‘1.85’.

If returning multiple variables use “in” instead of “=”.

* Create table members( mid number(2) Primary Key, Name Char(20)).
* Create Table book( Bid number(3), title Char(20)).
* Create table book issue(members\_ID number(2), References members, Book ID number (3) references book)).
* Primary key uniquely identifies the reference

Data dictionaries:

* ALL: Tables, Views, Indexes
* USER: Tables, Views, Indexes
* DBA: Tables, Views, Indexes
* User\_constraints, shows all user applied tables with constraints
* Select constraint\_type, table\_name, constraint\_name from [table name] SYSC00214. Where table-name
* TO remove a constraint: alter table [name] drop constraint SYSW00214. You’ll have to go into the data dictionary to find the name.
* Alter table [name] add constraint [constraint name]
* Normal convention for a constraint name is table name\_column\_constraint. Eg: tesco\_price\_tesc1.
* To remove the example constraint: “alter table tesco drop constraint tesc1”
* Views, contains information about the views. Tables about the tables and indexes about the indexes.
* Select \* user\_cons\_columns

26/9: Retrieve information from multiple tables:

Select from personal, results (add names of however many tables)

* Create personal table, results table
* Select from personal, results

Retrieving information from multiple tables is called joins. E.g. 1st row of table 1 is joined with 1st row of table 2. Ensure not to join the specified row from table 1 with all rows in the corresponding table.

In this case the two tables will be merged.

* Select personal.regno, name, results, subject, results, marks, from personal where personal.regno = results.regno and personal.regno = 1;
* Displays all records with the RegNo is 1 as the result of the above query is true in this case, and false for where these = 2.

Where there columns from either table with the same name you have to specify in the query which one e.g. Regno appears twice, so to refer to either you’d have to say personal.regno and results.regno.

Select x.regno, x.name, y.name. r.object(?). p.marks from personal x, results y, where x.regno = y.regno;

Select from Personal, Results, Salary:

And now, something else entirely:

Consultants example: How much salary is being paid to which department –

* Select sum(salary) from consultants by client group by client - will arrange the fields by the client column into groups on the basis of their similar client name. It will then sum the salaries in the group (from the sub query).
* TO change the heading: select sum(salary) as salary, client from consultants group by client. If you put subject after client it will give an error as the name subject has not been included in the initial group clause.

To make sub-groups:

* Create a group on the basis of the client.
* The group will then be divided into Sub-Groups on the basis of Sub-Groups.
* You should then have 4 sub-groups in this table (…?)

Group by “Regno”:

* Select sum(salary), regno from consultants group by regno;

Select department\_id from employees group by department\_id having count(\*)>15; The Department’s which have more than 15 people in them.

Select department\_id, count(\*) from employees where department\_id in (20, 30, 10) group by department\_id having count(\*)>15; the where clause will filter the records that satisfy the condition, group will group them. Then the having condition will apply **only** on the groups that have been made (10, 20, 30).

Select from employees, departments where employees.department\_id = departments.department\_id.

Select first\_name, salary, department\_name from employees, departments where employees.department\_id = departments.department\_id; retrieves the names of the departments.

Select count(\*), sum(salary), department\_name from employees, departments where employees.department\_id = departments.department\_id group by departments.department\_id;

Back to the sales example:

Working out the Quantity of 7Ups sold:

* Select sum(sales.quantity) from sales where pid = 100;

Bank Example:

Acc No:

Char 1: Acc Type: C – Current, S- Savings Ch 2: Country Code E- Eng, S – Sco, W – Wal – I -Ire. Char’s 3, 4, and 5: 001 (Account code)

E.g:

- CE001, CW002, SW001 etc

Query to create the AccNo: select max (substr(AccNo, 3, 3)) + 1 where substrAccNo, 1, 1 = ‘s’ to generate the 3 characters at the end of the AccNo.

To enter a first record: select nvl(max (substr(AccNo, 3, 3)), 0) + 1;

The L Pad (left pad) and R Pad (right pad) functions:

* Select R Pad(‘ABC’, ‘y’, ‘x’) from bank

Working equation:

Insert into bank values (select( ‘S’||’E’||lpad(nvl(max (substr(AccNo, 3, 3)), ***0***) + 1, 3, 0) from bank when substr(accno, 1, 1)=’s’), ‘Peter’);

***0:***The number you change to alter the end digit. (And the name at the end obviously).

Creating an index example:

An index in this case is an alphabetical reference file for the specified table.

* Create index schoolidx on school.regno
* Select \* from school where regno = 10000
* Creating indexes make searches more efficient
* You are not able to see the indexes
* When using the “where” clause it will use the index
* Sometimes you have to create multiple indexes depending on how many basis’ you need to search, however indexes are only worth making on a basis which is unique, such as NI number, employee number, passport number (person specific numbers).

Creating a view:

* Create view school view is select \* from school;
* Select \* from school view;
* Insert into school view values …
* Doesn’t actually store values in the table, it just displays it in the view.
* Essentially a “print preview mode”, or a sample table for those who need to see a view of the table
* The view cannot store the data
* Create view, itdept is select regno, name, email from school;
* Will display specified information by the admin for those who have been given specified access to it.
* Views can be made read only
* Views work the same way tables do.

More views:

* Create view topposition is select name from school \* where marks = (select max(marks) from school);

Creating users and privileges:

* Connect; connect with a different user
* System and object privileges in Orcale.
* Systems defines the actions that the user can do, creating tables, connecting etc
* Object privileges define what objects the user can use.
* Privileges can be given with the command “grant”
* E.g. “grant create table to Shafeeq;”. “Grant create view, create index, create session to My Friend;”
* “Grant create table to Peter, Peter and Peter” to create rights to multiple user
* “Grant create index to public” to give public rights to certain functions.
* Role: a collection of privileges – “Create role My Friends, grant create user, create table to my friends, create view to my friends”.
* This larger role “My Friends” can now be assigned to a user, e.g. “Grant My Friends to Shafeeq” saving time from creating individual roles each time.
* “Revoke” removes rights from users;
* “Revoke create user from My Nemesis”.
* When creating a user grant them “resources” a bulk collection of privileges
* DBA – database admin, contains all the privileges. Can also be granted to other users.
* You can have multiple DBAs.
* You can grant select rights on select tables to select users, eg: “grant select on college to my friend” or “grant select, insert on college to my friend”.
* For them see their approved version of this table: “select \* from shafeeq.college”
* Data dictionaries help find whether a name is a user or a role.

Transactions:

* An change of data in the database (e.g. change, delete, update).
* Select is not a transaction.
* You can create multiple sessions with different user
* Can only make a change in the database via the DML
* My friend
* You can save or reverse transactions
* To save “write commit” and store them permanently
* You can also roll back the transaction and reverse it. It will affect the whole transaction by the way to its previously saved state.
* Both Commit and Roll Back will close the transaction.
* When you close an active session your transaction will be Committed, then your session will be closed.
* DDL also commits your transaction.
* Using Save Point commands (eg. Save Points A, B and C) you can Roll Back to one of the set Save Points.

Set Auto Commit On:

* Starts transaction, sets the command in the database, and executes the command in the database.
* V\_transaction: see current open transactions.