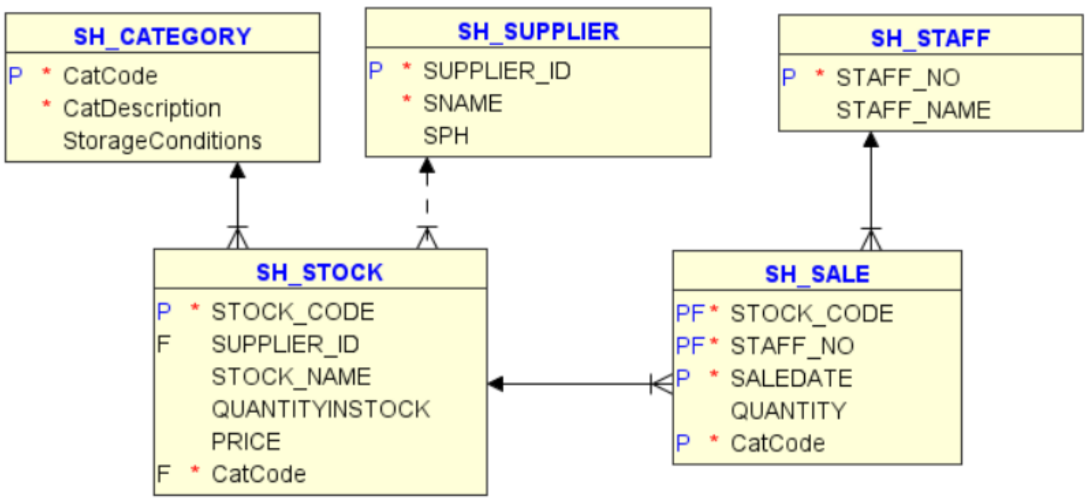
DB2 Lab 2 Advanced queries, corner shop

1. Log on to your own role on the University PostgreSQL server (you need to use the desktop – you cannot access it using WiFi), using the following credentials:
    - Host: 147.252.250.51
    - Username: <Your student number>
    - Password: <Your student number>

   You will be logged on to the postgres database. Clicking the arrow beside the postgres database, expand it and the schemas. Scroll down until you get to your student number, which should be in bold. From there, right click, click SQL Editor > SQL Console.
2. Change your password. From the SQL console, use the following command to change your password. The following example assumes a student number of G99999999, where you want a new password of MyNewPwd. Whatever new value you use, DON'T FORGET IT!
   ```
   Alter role "G99999999" with password 'MyNewPwd';
   ```

3. Check your access to the tables in this week's example. The tables are created and populated on the server in Central Quad. They are in the SAMPLE schema. You have been given access to the schema. To access the tables, open a SQL window from your own login. To access the tables in another schema (in this case "sample") you can either:
    - Change your search path
      ```
      SET SEARCH_PATH = "sample";
      ```
      --To test your connection, select * from each of the tables. However, you cannot create views in that path. To create a view, set your search path back to your own account.
    - From your own schema, prefix each table name with schema name, "sample". (the dot is also required).
      To create views referencing another schema, run something like the following:
      ```
      create or replace view STAFFSOLD as
      select STAFF_NAME, STOCK_NAME
      from "sample".SH_STAFF
      join "sample".SH_SALE USING (STAFF_NO)
      JOIN "sample".SH_STOCK USING (STOCK_CODE);
      ```



A local business called the Corner Shop acts as a newsagent and general shop to the public. There are a few staff members working there. Each staff member is allocated a unique staff_no when they join the shop and their staff no is recorded, with their name, in the SH_STAFF table. There is a wide variety of stock and most of the stock comes from recorded suppliers. Stock is given a category, so that it can be properly stored. When stock comes into the shop, if it is not stock that was sold previously, it is given a unique stock code. The stock_code, stock_name and price for each stock item is recorded in the STOCK table, along with the quantity (quantityinstock). It is given a CatCode to show its category from the SH_CATEGORY. Category descriptions include 'Stationery', 'Long-life food', 'refrigerated food' and 'heavy items'. If the stock came in from a regular supplier, that supplier's supplier_id is recorded in the SH_STOCK table. However, sometimes the shop sells items such as homemade cakes, or flags or bunting for some event. These do not have a supplier_id. Regular suppliers have a unique supplier_id, a name (sname) and a phone number (sph) that are recorded in the SH_SUPPLIER table. When a sale is made, the stock_code and quantity is recorded in the SH_SALE table, along with the STAFF_NO of the staff member who made the sale, and the date and time of the sale.

TABLES:

- SH_SUPPLIER (SUPPLIER_ID(PK), SNAME, SPH)
- SH_STAFF(STAFF_NO(PK), STAFF_NAME)
- SH_CATEGORY (CatCode(PK), CatDescription, StorageConditions)
- SH_STOCK(STOCK_CODE(PK), SUPPLIER_ID(FK optional), STOCK_NAME, QUANTITYINSTOCK, PRICE, CatCode)
- SH_SALE(STOCK_CODE(PK, FK), STAFF_NO(PK, FK), SALEDATE(PK), QUANTITY)

RELATIONSHIPS:

- SH_SUPPLIER: SH_STOCK 0..1:0..many using SUPPLIER_ID , non-identifying
- SH_CATEGORY:SH_STOCK 1:0..many using CatCode, non-identifying
- SH_STAFF: SH_SALE 1:0..many using STAFF_NO, identifying
- SH_STOCK: SH_SALE 1:0..many using STOCK_CODE, identifying

NOTE: An identifying relationship means that the foreign key also acts as part of the primary key. In this case, the SH_SALE primary key is a combination of STOCK_CODE, STAFF_NO and SALEDATE. SALEDATE is used so that the same staff member can sell the same stock item multiple times.

Write SQL to do the following:

1. Name staff members who sold notepads.
2. Name staff members that sold nothing.
3. Name staff members that sold both notepads and chocolate bars.
4. Name staff members that sold either notepads or chocolate bars, but not both.
5. Name staff members that only sold notepads.
6. Name staff members that have sold all stock.
7. Name staff members that have sold all stock supplied by 'Cadbury'.

The next queries use windows functions. As they are averaging money values, run the following code:

```
set lc_monetary to "en_IE.utf8";
```

This sets the money currency to euro. NOTE: If this doesn't work, it'll show $, which is fine too. To show a money attribute (e.g. price)

```
select cast(price as money) "Cost" from sh_stock;
```

8. Using a windows function, select the stock_name, category description (CatDescription), and price of each stock item, and average price of stock items in that category.
9. Using the 'lag' function, display each stock name, its category description, price, the price of the previous stock item in that category (ordered by price) and the difference between the price of the previous item and of this item. See 'Sample Windows Functions.sql' and this week's slides (Slide headed 'LAG') for inspiration.

Marking will be as follows:

1 mark for completing and explaining a query from the list 1 to 7, chosen by the lab supervisor.
1 mark for completing and explaining a query from 8 and 9, chosen by the lab supervisor.