

ASSIGNMENT 5: SQL

Francis Eseko

ESKFRA001

The Queries are as listed below, with the corresponding code and results obtained when running.

- 1) Show all information in the **offices** relation.

SQL code: `select * from offices;`

The screenshot shows a database query tool interface. The query editor at the top contains the SQL code `select * from offices;`. Below the editor, the 'Result Grid' displays the data from the 'offices' table. The table has 10 columns: officeCode, city, phone, addressLine1, addressLine2, state, country, postalCode, and territory. There are 7 rows of data, with the first row highlighted. The last row shows NULL values for all columns.

| | officeCode | city | phone | addressLine1 | addressLine2 | state | country | postalCode | territory |
|---|------------|---------------|------------------|--------------------------|--------------|------------|-----------|------------|-----------|
| 1 | 1 | San Francisco | +1 650 219 4782 | 100 Market Street | Suite 300 | CA | USA | 94080 | NA |
| 2 | 2 | Boston | +1 215 837 0825 | 1550 Court Place | Suite 102 | MA | USA | 02107 | NA |
| 3 | 3 | NYC | +1 212 555 3000 | 523 East 53rd Street | apt. 5A | NY | USA | 10022 | NA |
| 4 | 4 | Paris | +33 14 723 4404 | 43 Rue Jouffroy D'abbans | NULL | NULL | France | 75017 | EMEA |
| 5 | 5 | Tokyo | +81 33 224 5000 | 4-1 Koicho | NULL | Chiyoda-Ku | Japan | 102-8578 | Japan |
| 6 | 6 | Sydney | +61 2 9264 2451 | 5-11 Wentworth Avenue | Floor #2 | NULL | Australia | NSW 2010 | APAC |
| 7 | 7 | London | +44 20 7877 2041 | 25 Old Broad Street | Level 7 | NULL | UK | EC2N 1HN | EMEA |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

- 2) Show any one tuple in the **payments** relation (just one, no more).

SQL code: `select * from payments limit 1;`

The screenshot shows a database query tool interface. The query editor at the top contains the SQL code `select * from payments limit 1;`. Below the editor, the 'Result Grid' displays the data from the 'payments' table. The table has 4 columns: customerNumber, checkNumber, paymentDate, and amount. There is 1 row of data, with the first row highlighted. The last row shows NULL values for all columns.

| | customerNumber | checkNumber | paymentDate | amount |
|---|----------------|-------------|---------------------|---------|
| 1 | 103 | HQ336336 | 2004-10-19 00:00:00 | 6066.78 |
| * | NULL | NULL | NULL | NULL |

- 3) Show how many tuples there are in the **orders** relation.

SQL code: `select count(*) as 'tuplenuumber' from orders;`

Results: 326 tuples

ables

```

1 • select * from offices;
2 • select * from payments limit 1;
3 • select count(*) as 'tuplenunder' from orders;

```

Result Grid | Filter Rows: | Export: | Wrap

| tuplenunder |
|-------------|
| 326 |

- 4) Show all **employees** tuples where **reportsTo** is the same as **employeeNumber**.

SQL code: select * from employees where employeeNumber=reportsTo;

les

```

1 • select * from offices;
2 • select * from payments limit 1;
3 • select count(*) as 'tuplenunder' from orders;
4 • select * from employees where reportsTo=employeeNumber;
5 • select * from payments where amount>100000 order by amount desc;
6 • select * from employees where employeeNumber=1188 or employeeNum

```

Limit to 1000 rows

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

| employeeNumber | lastName | firstName | extension | email | officeCode | reportsTo | jobTitle |
|----------------|----------|-----------|-----------|-------|------------|-----------|----------|
|----------------|----------|-----------|-----------|-------|------------|-----------|----------|

- 5) Show all information in the **payments** relation for payments exceeding 100 000, in decreasing order (i.e. from highest payment downwards).

SQL code: select * from payments where amount>100000 order by amount desc;

SQL code:

```

1 select * from offices;
2 select * from payments limit 1;
3 select count(*) as 'tuplenumber' from orders;
4 select * from employees where reportsTo=employeeNumber;
5 select * from payments where amount>100000 order by amount;
6 select * from employees where employeeNumber=1188 or employeeNumber=1504;

```

Result Grid

| customerNumber | checkNumber | paymentDate | amount |
|----------------|-------------|---------------------|-----------|
| 141 | JE105477 | 2005-03-18 00:00:00 | 120166.58 |
| 141 | ID10962 | 2004-12-31 00:00:00 | 116208.4 |
| 124 | KI131716 | 2003-08-15 00:00:00 | 111654.4 |
| 148 | KM172879 | 2003-12-26 00:00:00 | 105743 |
| 124 | AE215433 | 2005-03-05 00:00:00 | 101244.59 |
| NULL | NULL | NULL | NULL |

- 6) Show all information in the **employees** relation for **employeeNumbers** 1188 and 1504.

SQL code: select * from employees where employeeNumber=1188 or employeeNumber=1504;

SQL code:

```

2 select * from payments limit 1;
3 select count(*) as 'tuplenumber' from orders;
4 select * from employees where reportsTo=employeeNumber;
5 select * from payments where amount>100000 order by amount desc;
6 select * from employees where employeeNumber=1188 or employeeNumber=1504;
7 select productCode, ROUND(buyPrice*1.14, 2) as 'totalPrice' from products where quantityInStock<100;

```

Result Grid

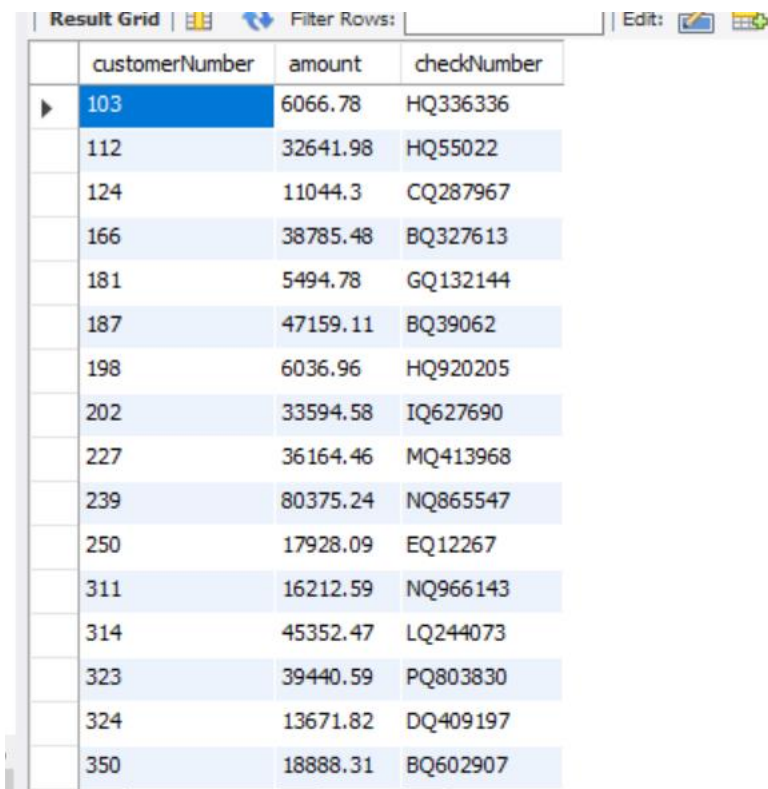
| employeeNumber | lastName | firstName | extension | email | officeCode | reportsTo | jobTitle |
|----------------|----------|-----------|-----------|--------------------------------|------------|-----------|-----------|
| 1188 | Firrelli | Julie | x2173 | jfirrelli@classicmodelcars.com | 2 | 1143 | Sales Rep |
| 1504 | Jones | Barry | x102 | bjones@classicmodelcars.com | 7 | 1102 | Sales Rep |

- 7) Show the **productCode** of all **products** having their **quantityInStock** below 100, along with their total price. The total price is the **buyPrice** plus VAT (VAT is 14% of **buyPrice**).

SQL code: select productCode, ROUND(buyPrice*1.14, 2) as 'totalPrice' from products where quantityInStock<100;

- 11) Show the **customerNumber** and **amount** for all **payments** with a 'Q' as the 2nd character of the **checkNumber** (a check is a cheque!)

SQL code: select customerNumber, amount, checkNumber from payments where checkNumber like '_Q%';

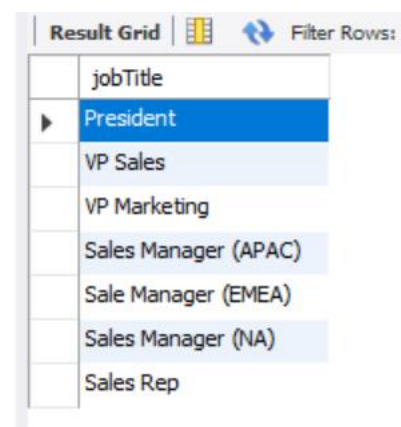


The screenshot shows a 'Result Grid' window with a table of payment records. The table has three columns: 'customerNumber', 'amount', and 'checkNumber'. The records are filtered to show only those where the check number starts with 'Q'. The first row is highlighted in blue.

| customerNumber | amount | checkNumber |
|----------------|----------|-------------|
| 103 | 6066.78 | HQ336336 |
| 112 | 32641.98 | HQ55022 |
| 124 | 11044.3 | CQ287967 |
| 166 | 38785.48 | BQ327613 |
| 181 | 5494.78 | GQ132144 |
| 187 | 47159.11 | BQ39062 |
| 198 | 6036.96 | HQ920205 |
| 202 | 33594.58 | IQ627690 |
| 227 | 36164.46 | MQ413968 |
| 239 | 80375.24 | NQ865547 |
| 250 | 17928.09 | EQ12267 |
| 311 | 16212.59 | NQ966143 |
| 314 | 45352.47 | LQ244073 |
| 323 | 39440.59 | PQ803830 |
| 324 | 13671.82 | DQ409197 |
| 350 | 18888.31 | BQ602907 |

- 12) What **jobTitles** exist in the database?

SQL code: select distinct(jobTitle) from employees;



The screenshot shows a 'Result Grid' window with a table of distinct job titles. The table has one column: 'jobTitle'. The records are listed in descending order of the number of employees holding that title. The first row is highlighted in blue.

| jobTitle |
|----------------------|
| President |
| VP Sales |
| VP Marketing |
| Sales Manager (APAC) |
| Sale Manager (EMEA) |
| Sales Manager (NA) |
| Sales Rep |

- 13) Show **productName** and **buyPrice** of the product(s) with the highest **buyPrice**.

SQL code: select productName, Max(buyPrice) as buyingPrice from products;

| Result Grid | | Filter Rows: | Export |
|-------------|---------------------------------------|--------------|--------|
| | productName | buyingPrice | |
| ▶ | 1969 Harley Davidson Ultimate Chopper | 103.42 | |

- 14) Show **orderNumber**, **status**, **quantityOrdered** and **productName** for all **products** from **productVendor** 'Exoto Designs' that have **status** 'Cancelled'.

SQL code: select o.orderNumber, o.status, pr.quantityOrdered, p.productName from orders as o, products as p, orderdetails as pr where pr.orderNumber=o.orderNumber and p.productCode=pr.productCode and o.status='Cancelled' and p.productVendor='Exoto Designs';

| Result Grid | | Filter Rows: | Export: | Wrap Cell Content: |
|-------------|-------------|--------------|-----------------|---|
| | orderNumber | status | quantityOrdered | productName |
| ▶ | 10167 | Cancelled | 34 | 1941 Chevrolet Special Deluxe Cabriolet |
| | 10179 | Cancelled | 45 | 1997 BMW F650 ST |
| | 10248 | Cancelled | 42 | 1904 Buick Runabout |
| | 10260 | Cancelled | 23 | 1992 Porsche Cayenne Turbo Silver |
| | 10262 | Cancelled | 34 | 1941 Chevrolet Special Deluxe Cabriolet |


- 15) Show the **productCode** of all **products** that have never been ordered.

SQL code: select p.productCode from products as p left join orderdetails as pr on pr.productCode=p.productCode where pr.productCode is null;

| Result Grid | |
|-------------|-------------|
| | productCode |
| ▶ | S18_3233 |

- 16) Show how many **employees** there are in each office (give **officeCode** and value each time).

SQL code: select o.officeCode, count(e.officeCode) as 'numOccupants' from offices as o, employees as e where e.officeCode=o.officeCode group by o.officeCode;

| | | | |
|-------------|------------|---|-----------------------------------|
| Result Grid | |  | Filter Rows: <input type="text"/> |
| | officeCode | numOccupants | |
| ▶ | 1 | 6 | |
| | 2 | 2 | |
| | 3 | 2 | |
| | 4 | 5 | |
| | 5 | 2 | |
| | 6 | 4 | |
| | 7 | 2 | |



- 17) Show how many **customers** each employee is associated with (as **salesRepEmployeeNumber**), but only for employees who are the **salesRepEmployeeNumber** for at least 1 customer. Give **employeeNumber** and value each time.

SQL code: select e.employeeNumber, count(c.salesRepEmployeeNumber) as 'numCustomers' from employees as e, customers as c where c.salesRepEmployeeNumber=e.employeeNumber group by e.employeeNumber;

variables

5

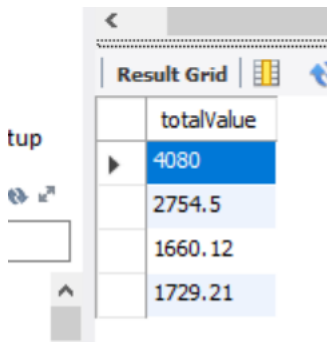
Setup

| | employeeNumber | numCustomers |
|---|----------------|--------------|
| ▶ | 1165 | 6 |
| | 1166 | 6 |
| | 1188 | 6 |
| | 1216 | 6 |
| | 1286 | 7 |
| | 1323 | 8 |
| | 1337 | 6 |
| | 1370 | 7 |
| | 1401 | 10 |
| | 1501 | 8 |
| | 1504 | 9 |
| | 1611 | 5 |
| | 1612 | 5 |
| | 1621 | 5 |
| | 1702 | 6 |

- 18) What was the total value of **orderNumber** 10100 i.e. the total of (**quantityOrdered** * **priceEach**) over all its orderlines?

SQL code: `select (pr.quantityOrdered*pr.priceEach) as 'totalvalue' from orderdetails as pr where pr.orderNumber='10100';`

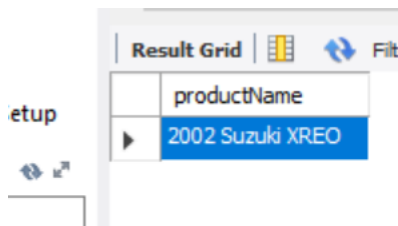


The screenshot shows a 'Result Grid' window with a table containing the results of the SQL query. The table has one column labeled 'totalValue' and four rows of data. The first row is highlighted in blue.

| totalValue |
|------------|
| 4080 |
| 2754.5 |
| 1660.12 |
| 1729.21 |

- 19) Show the **productName** of the product/s with the largest **quantityInStock**.

SQL code: `select productName from products where quantityInStock = (select Max(quantityInStock) from products) group by productName;`



The screenshot shows a 'Result Grid' window with a table containing the results of the SQL query. The table has one column labeled 'productName' and one row of data, which is highlighted in blue.

| productName |
|------------------|
| 2002 Suzuki XREO |

- 20) Show the **employeeNumber** of **employees** who **reportsTo** the same person as does **employeeNumber** 1313 (i.e. who have the same boss as 1313).

SQL code: `select e.employeeNumber from employees as e, (select * from employees as ee where ee.employeeNumber=1313) as bb where e.reportsTo=bb.reportsTo;`

- 21) Show the **employeeNumber** of all **employees** who are superiors of **employeeNumber** 1313 (i.e. the person 1313 **reportsTo**, and the employee who that person **reportsTo**, ... all the way up)

- 22) Devise a useful query of your own involving the most interesting usage of SQL you can think of.

Explain clearly in a comment what it is meant to find from the database. Also explain how you know the SQL for this query is correct (showing intermediate results if necessary). Marks here will be proportional to the complexity and usefulness of the query you implement.

- 23) Add a new **office** to the database, giving it **officeCode** 999 (meaning planned for later). This office will be in Cape Town, but no other details are known yet. Make **state** 'Western Province'.

SQL code: `insert into offices (officeCode, city , state) values(999, 'Cape Town', 'Western Cape');`

- 24) Employee 1313 is superstitious. Change their employee number in the database, giving them the employee number 1 greater than the largest employee number in the database.
- 25) **OrderNumber** 10101 was never signed by the customer. Remove it from the database.